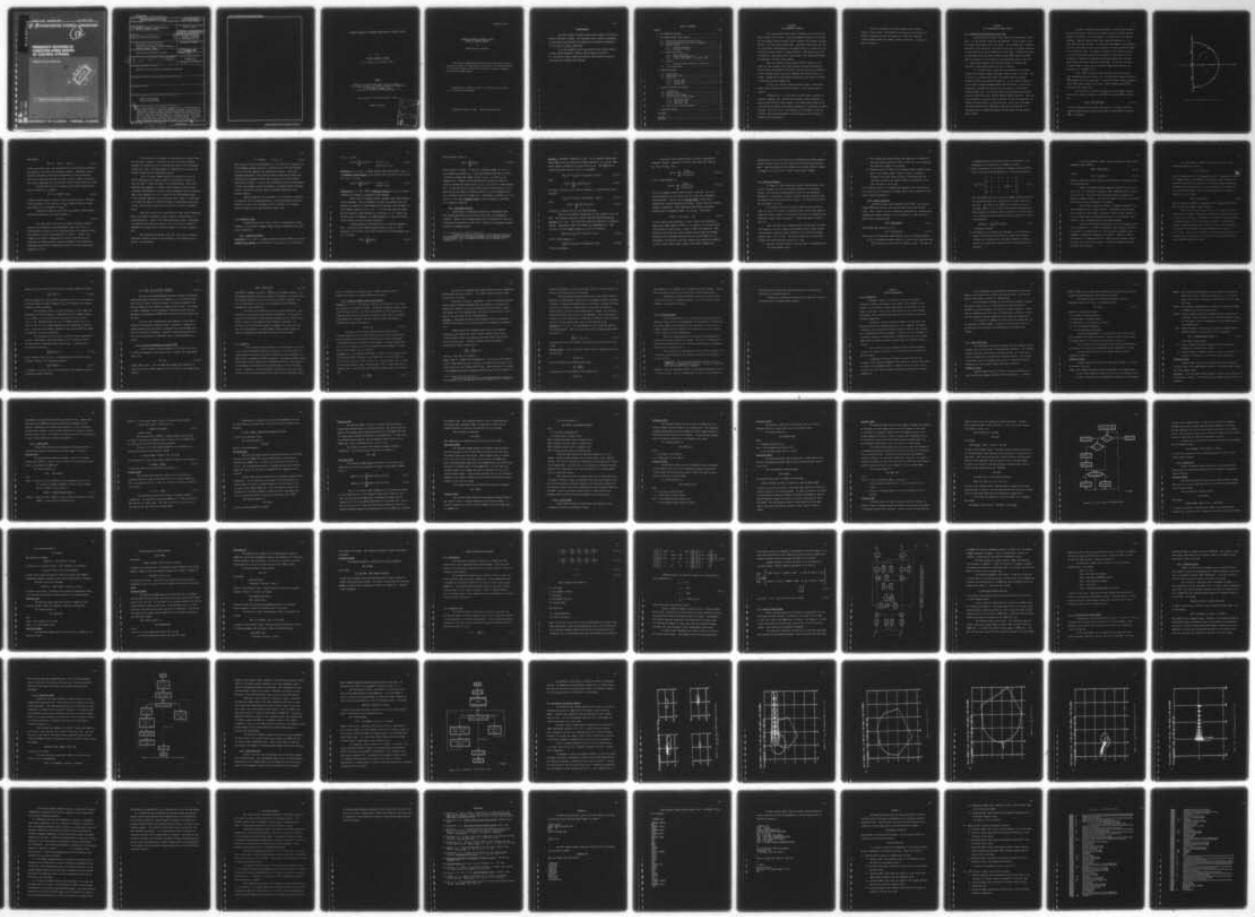
AD-A040 762 ILLINOIS UNIV AT URBANA-CHAMPAIGN COORDINATED SCIENCE LAB F/G 9/2
FREQUENCY METHODS IN COMPUTER AIDED DESIGN OF CONTROL SYSTEMS.(U)
DEC 76 S C SPIELMAN DAAB07-72-C-0259
UNCLASSIFIED R-753 NL

1 OF 2
AD A040762



REPORT R-753 DECEMBER, 1976 UILU-ENG 76-2241

CSL COORDINATED SCIENCE LABORATORY

12 B.S.

FREQUENCY METHODS IN COMPUTER AIDED DESIGN OF CONTROL SYSTEMS

STEPHEN CHRISTIE SPIELMAN

UNIVERSITY OF ILLINOIS - URBANA, ILLINOIS

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE *(and Subtitle)* (6) FREQUENCY METHODS IN COMPUTER AIDED DESIGN OF CONTROL SYSTEMS,		5. TYPE OF REPORT & PERIOD COVERED Technical Report
		6. PERFORMING ORG. REPORT NUMBER (14) R-753, UILU-ENG-76-2241
7. AUTHOR(s) (10) Stephen Christie/Spielman		8. CONTRACT OR GRANT NUMBER(s) (15) DAAB-07-72-C-0259; AF-AFOSR 73-2570-73
9. PERFORMING ORGANIZATION NAME AND ADDRESS Coordinated Science Laboratory ✓ University of Illinois at Urbana-Champaign Urbana, Illinois 61801		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Joint Services Electronics Program		12. REPORT DATE (11) December, 1976
		13. NUMBER OF PAGES 111
14. MONITORING AGENCY NAME & ADDRESS *(if different from Controlling Office)* (9) Master's thesis,		15. SECURITY CLASS. *(of this report)* UNCLASSIFIED
(12) 117p.		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY NOTES

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

Computer-Aided Design
Control System Design

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

State space methods have served an important role in the development of modern control theory. They were originally proposed to help solve the increasingly complex aerospace control problems which frequency response methods of the time could not deal with. Historically this theory has lead to a much greater understanding of the general nature of a system and to the solution of a great variety of control problems. These means have further been applied to many other areas in the sciences, thus firmly establishing the importance of modern system studies.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

097 700

FREQUENCY METHODS IN COMPUTER AIDED DESIGN OF CONTROL SYSTEMS

BY

STEPHEN CHRISTIE SPIELMAN

B.S., University of Illinois, 1974

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Electrical Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 1977

Thesis Adviser: Professor William R. Perkins

Urbana, Illinois

UILU-ENG 76-2241

FREQUENCY METHODS IN COMPUTER AIDED DESIGN OF CONTROL SYSTEMS

by

Stephen Christie Spielman

This work was supported in part by the Joint Services Electronics Program (U.S. Army, U.S. Navy and U.S. Air Force) under Contract DAAB-07-72-C-0259 and in part by the Air Force Office of Scientific Research under Contract AFOSR 73-2570.

ACKNOWLEDGMENT

The author wishes to express a most sincere thanks to his advisor, Professor William R. Perkins. The invaluable help, constant encouragement, and the pleasure and privilege of working with him during the preparation of this thesis is greatly appreciated.

He is also grateful to Mrs. Rose Harris for her superb typing of a rather untidy manuscript. Finally, special mention is given to Ms. Lynn Waverly, without whose ever presence and diversion my education would have been completed some time ago.

TABLE OF CONTENTS

CHAPTER 1
AN ALTERNATIVE APPROACH

State space methods have served an important role in the development of modern control theory. They were originally proposed to help solve the increasingly complex aerospace control problems which frequency response methods of the time could not deal with. Historically this theory has lead to a much greater understanding of the general nature of a system and to the solution of a great variety of control problems. These means have further been applied to many other areas in the sciences, thus firmly establishing the importance of modern system studies.

While the elegance of these methods cannot be denied, for the linear and time invariant system they pose many practical difficulties. These include incorporation of simplicity, low sensitivity, and robustness of the resulting control, as well as integrity and fault tolerance, in a design. Working with an input-output description of the plant is believed to solve many of these problems [1].

In fact, the classical frequency domain design of single input single output systems has never been replaced by state space methods in practice.

Rosenbrock et al., at the Control Systems Centre, University of Manchester, England, has extended these frequency domain techniques to multiple input multiple output systems. This design method, which he has named the inverse Nyquist array method, encompasses a large number of control problems. One of its fortes is the ability of the designer to shape the outcome of the resulting designs to suit his needs and still achieve an adequate system performance.

How this is possible will become evident in what follows. Chapter 2 relies heavily on Rosenbrock's own exposition of the method [2]. Chapter 3 presents an outline of programs to be used for interactive frequency domain design with computer graphics facilities. An example is given in Chapter 4.

CHAPTER 2
THE INVERSE NYQUIST ARRAY METHOD

2.1. Introducing Specifications and the Plant

The design process begins with an input-output description of the plant. Let the plant be linear and time invariant. It has an (ℓx1) vector $\bar{u}$ as input, and an (mx1) vector $\bar{y}$ as output. In the design, one is primarily interested in the response at each output of the plant as we apply sinusoidal forcing functions of various frequencies to each input. Therefore the plant is represented by an m x ℓ transfer function matrix $\overline{G(s)}$, which is rational, with the exception of the inclusion of pure exponential terms of the form e^{sT}. These terms represent pure time delay and may be included when necessary, as many plants exhibit this sort of behavior.

It is possible that a state space representation of the plant, arising from a physical model or available empirical data, is at hand. This, of course, is a suitable description of the plant after the required transfer functions are obtained via analysis. As one's actual interest is in the complex frequency mapping between input and output, it may be more appropriate to measure this data on the actual plant, if the physical situation allows this. For the type plants being considered the data may always be fitted into such a transfer function matrix expression. While this is not implicitly a necessary task to perform, it shall be assumed that the transfer function matrix is of the above form. In the case of designing from actual measured data, $\overline{G(s)}$ merely represents this data and is a convenient means to interpolate between the data points for the physical model at hand.

As usual in frequency domain techniques, one deals with the map of the open loop transfer function in the complex plane as a suitably defined contour is traversed. The encirclements of a critical point then indicate stability. Here the contour, which shall always be denoted by D, is in every case traversed clockwise. It consists of that segment of the imaginary axis from -jR to +jR, as well as the semicircle of radius R about the origin in the right half plane. The contour is indented to the left of any pole or zero of the transfer function at the origin or on the $j\omega$ axis, as one in general is interested in asymptotically stable systems and, therefore, must include these points within the closed path. The radius R is taken to be large enough to insure that all right half plane poles and zeros are enclosed by D. See Figure 2.1.1.

Given a $\overline{G(s)}$, consider a design procedure based on the closed loop configuration shown in Figure 2.1.2. An (ℓXk) input compensation matrix $\overline{K(s)}$ relates the error $\overline{E}$ with input $\overline{U}$. Similarly an (kXm) output compensator matrix $\overline{L(s)}$ relates output $\overline{Y}$ to $\overline{Z}$, while a (kXk) feedback matrix $\overline{F(s)}$ is inserted in the return path.

For notational simplicity, the (kXk) open loop transfer function matrix of the system relating the error $\overline{E(s)}$ to the output $\overline{Z(s)}$ is defined as $\overline{Q}$

$$\overline{Q(s)} = \overline{L(s)}\ \overline{G(s)}\ \overline{K(s)}. \tag{2.1.1}$$

As one is interested in the closed loop responses, the (kXk) closed loop transfer function matrix relating inputs $\overline{V(s)}$ to outputs $\overline{Z(s)}$ is defined as $\overline{H(s)}$. By assuming

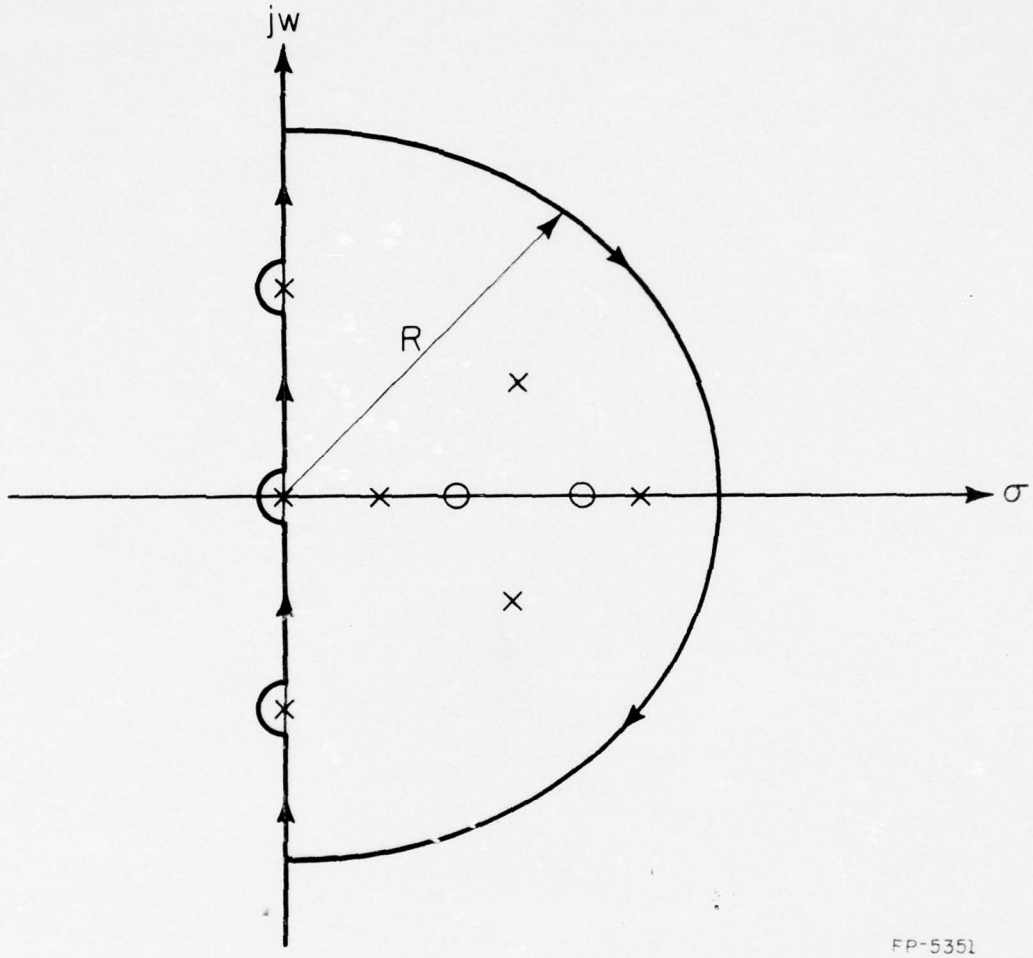


Figure 2.2.1. The Nyquist contour in the complex plane.

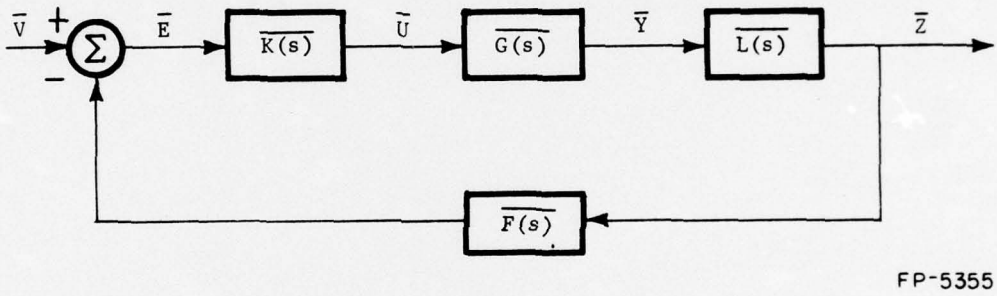


Figure 2.1.2. System block diagram.

$$|\bar{I}_k + \overline{Q(s)}\ \overline{F(s)}| \not\equiv 0 \tag{2.1.2}$$

it immediately follows that

$$\bar{H}(s) = (\bar{I}_k + \overline{Q(s)}\ \overline{F(s)})^{-1}\ \overline{Q(s)}. \tag{2.1.3}$$

In assuming

$$|\bar{I}_k + \overline{F(s)}\ \overline{Q(s)}| \not\equiv 0 \tag{2.1.4}$$

the closed loop transfer function may be expressed as

$$\overline{H(s)} \quad \overline{Q(s)}(\bar{I}_k + \overline{F(s)}\ \overline{Q(s)})^{-1}. \tag{2.1.5}$$

As both (2.1.3) and (2.1.5) show that $\bar{H}$ is a function of $\bar{Q}$, $\bar{F}$, and the Laplace variable s, one might consider

$$\bar{H} = \bar{H}[\bar{Q},\bar{F},s] \tag{2.1.6}$$

as a compact way of expressing this closed loop transfer function matrix. While this represents a relatively simple expression in terms of computational considerations, it shows very little in terms of relating the open loop system to the closed loop system. As our interest is in designing for an acceptable closed loop system performance, a further investigation into a possible means to these ends.becomes necessary.

2.2. Inverse Relationships and their Consequences

The results in the previous section are valid provided that the determinants (2.1.2) and (2.1.4) exist. It should now be assumed that

$$|\bar{Q}| \not\equiv 0 \tag{2.2.1}$$

which implies

$$|\bar{K}| \neq 0, \quad |\bar{G}| \neq 0 \quad |\bar{L}| \neq 0. \tag{2.2.2}$$

If this were not the case, the system would not be functionally controllable and therefore, be of very little practical interest. Functional controllability, also known as functional reproducibility, loosely means that there exists some input which will generate a specific bounded output from zero initial conditions, for all time other than the initial time [3, p. 169].

Provided that (2.2.1) is satisfied, $\bar{Q}$ will have an inverse which shall be denoted by a circumflex.

$$\bar{Q}^{-1} = \hat{Q} = (\bar{L}\bar{G}\bar{K})^{-1} = \hat{K}\hat{G}\hat{L} \tag{2.2.3}$$

A similar notation shall be used for all other matrix inverses. Note that the (i,j) element of $\hat{Q}$ is referred to as $\hat{q}_{ij}$ whereas the inverse of that same element in Q is referred to as $(q_{ij})^{-1}$.

Using inverse matrices, the closed loop transfer function matrix implicitly defined in (2.1.6) has an inverse of

$$\hat{H} = \bar{F} + \hat{Q}. \tag{2.2.4}$$

This shows that if $\hat{Q}$ exists, the closed loop inverse transfer function can be expressed in the astoundingly simple form of (2.2.4). Being additive by nature, it is the convenient means of relating the closed loop system to the open loop system which is desired. Clearly, more insight can be found in working with the inverse system and the easy transition in closing the loops, than from working with the noninversed system and dealing with the nonintuitive nature of the matrix multipliers imbedded in (2.1.6).

In a design one is interested in controlling the k outputs $\bar{Z}$ with the respective k inputs $\bar{V}$. To do this, the open loop inverse transfer function $\hat{Q}$ is constructed so as to minimize the effect of the off diagonal entries in $\bar{G}$. Once a sufficient degree of noninteraction is present, the original problem has been considerably reduced.

It is, of course, always possible to construct $\hat{Q}$ so that the desired noninteraction is present simply by choosing $\hat{K}$ or $\hat{L}$ equal to $\bar{G}$. This will make $\hat{Q}$ diagonal and equal to the identity matrix. While such a method has been suggested, it imposes severe problems in terms of plant variations, implementation in terms of stable subsystems, and the resulting shape of the frequency loci [4]. Perhaps the most objectionable reason is the necessary complexity of the resulting control scheme. As simplicity in a design often makes the difference between an engineering success and a failure, one could never justify such a scheme as this with its inherent faults.

Quite the contrary, one in fact wishes to find a matrix compensator which is as simple as possible in order to achieve a required amount of noninteractiveness. Ideally, this compensating matrix would be independent of frequency and as sparse as possible. This is the easiest control to implement on the actual plant, and would probably be the most reliable as well.

When sufficient decoupling of the open loop system is attained, appropriate stability regions can be determined so that feedback may be applied. By considering

$$\overline{F} = \text{diag}\{f_i\} \ , \qquad i = 1,2,\ldots,k \tag{2.2.5}$$

where each f_i is scalar and independent of s, the closed loop frequency loci of the diagonal transfer functions in $\hat{H}$ become those of the corresponding loci in $\hat{Q}$ with the imaginary axis appropriately shifted. Single loop frequency compensation in the classical sense of phase-lag and phase-lead networks may be designed to modify these frequency loci for the desired input-output responses. In short, our multivariable control problem becomes k single loop designs, in which an abundance of frequency domain techniques exist [2, pp. 28-116], and more modern algorithmic methods have never entirely replaced.

Before showing how this is possible, it is necessary to formalize the intuitive idea of noninteractiveness. As the design means adopted are graphical in nature, this information must be included on the diagonal inverse frequency loci of each individual control loop.

2.3. Dominant Systems

Rosenbrock calls these loosely coupled or noninteractive systems dominant, as the principle diagonal element primarily determines that input-output relationship.

2.3.1. Diagonal Dominance

Definition 1 [2, p. 142]: A complex rational (k×k) matrix $\overline{Z(s)}$ is said to be diagonally row dominant on the contour D if $z_{ii}(s)$ has no pole on D for all

$i = 1,2,\ldots,k$, and

$$|z_{ii}(s)| - \sum_{\substack{j=1 \\ j\neq i}}^{k} |z_{ij}(s)| > 0 \qquad \forall i = 1,2,\ldots,k \text{ and for all } s \text{ on } D. \tag{2.3.1}$$

Definition 2 [2, p. 142]: A complex rational (k×k) matrix $\overline{Z(s)}$ is said to be diagonally column dominant on the contour D if $z_{ii}(s)$ has no pole on D for all $i = 1,2,\ldots,k$, and

$$|z_{ii}(s)| - \sum_{\substack{j=1 \\ j\neq i}}^{k} |z_{ji}(s)| > 0 \qquad \forall i = 1,2,\ldots,k \text{ and all } s \text{ on } D. \tag{2.3.2}$$

Definition 3 [2, p. 142]: $\overline{Z(s)}$ is diagonally dominant on D if it is either diagonally row dominant or diagonally column dominant.

Observe that row dominance implies dominance, and column dominance implies dominance. Then for dominance, the sum of the moduli of the off diagonal row elements or column elements must be greater than the modules of that diagonal element for all s on D. It is clear that rows and columns may not be mixed to check dominance at a particular s on D. Instead, one may have either row dominance, or column dominance as s takes on different values around D, and still satisfy Definition 3.

A simple graphical construction exists to check the dominance of $\hat{Q}(s)$. Let $\hat{q}_{ii}(s)$ map the familiar Nyquist contour D into Γ_i. For a particular s on D, draw a circle with its center at the corresponding point on Γ_i of radius

$$d_i(s) = \sum_{\substack{j=1 \\ j\neq i}}^{k} |\hat{q}_{ij}(s)| \tag{2.3.3}$$

which is based on rows, or

$$d_i'(s) = \sum_{\substack{j=1 \\ j\neq i}}^{k} |\hat{q}_{ji}(s)| \tag{2.3.4}$$

which is based on columns. Do this for a sufficient number of s on D so as to define an area about Γ_i. If the area defined by this set of circles of radius d_i excludes the origin, for all $i = 1,2,\ldots,k$, then $\hat{Q}$ is row dominant on D. Similarly if the area defined by the set of circles of radius d_i' excludes the origin for all $i = 1,2,\ldots,k$, then Q is column dominant on D. In this same sense, if none of the circles of either radius d_i or of radius d_i' has the origin as an interior point or on its circumference, for all i as before, then $\hat{Q}$ is dominant on D. These bands, as defined by (2.3.2) and (2.3.3) are called Gershgorin bands as their usefulness is dependent upon Gershgorin's Theorem.*

2.3.2. Ostrowski's Theorem

Systems which are diagonally dominant have determinants which show interesting properties and will be of use later in investigating stability. However, the following theorem, due to Ostrowski [5], which was rediscovered by Rosenbrock, allows for the location of the diagonal transfer function in a dominant system.

---

*Gershgorin's Theorem, while being of little practical value for this design method, has important consequences in the rigorous development of what follows. For its statement and proof, one is referred to the literature [3, p. 11].

Theorem 1: Ostrowski's Theorem [2,p. 149]: Let the rational complex (k×k) matrix $\overline{Z(s)}$ be row (or respectively column) dominant for $s = s_o$ on any simple closed contour C, having on it no pole of any $z_{ii}(s)$. Then $\overline{Z(s_o)}$ has an inverse $\hat{Z}(s_o)$ which, for all $i = 1,2,\ldots,k$ satisfies:

$$|\hat{z}_{ii}(s_o)^{-1} - z_{ii}(s_o)| < \phi_i(s_o)d_i(s_o) < d_i(s_o) \tag{2.3.5}$$

where

$$\phi_i(s_o) = \max_{\substack{j \\ j \neq i}} d_j(s_o)/|z_{jj}(s_o)| \tag{2.3.6}$$

and $d_i(s_o)$ is given by (2.3.3) for row dominance., or alternatively, which satisfies

$$|z_{ii}(s_o)^{-1} - z_{ii}(s_o)| < \phi_i'(s_o)d_i'(s_o) < d_i'(s_o) \tag{2.3.7}$$

where

$$\phi_i'(s_o) = \max_{\substack{j \\ j \neq i}} d_j'(s_o)/|z_{jj}(s_o)| \tag{2.3.8}$$

and $d_i'(s_o)$ is given by (2.3.4), for column dominance.

The proof of Ostrowski's theorem may be found elsewhere [3,pp. 204-206]. This theorem of course has several important design implications.

Theorem 2 [2, p. 150]: Let $\hat{Q}$ exist and satisfy the conditions of Ostrowski's Theorem. Furthermore, assume $\overline{F}$ is diagonal and independent of s. Then for each s on D the diagonal element h_{ii} of (2.1.6) satisfies

$$|h_{ii}^{-1}(s) - (f_i + \hat{q}_{ii}(s))| < \phi_i(s)d_i(s) < d_i(s) \tag{2.3.9}$$

if $\hat{H}$ is column dominant, or

$$|h_{ii}^{-1}(s) - (f_i + \hat{q}_{ii}(s))| < \phi_i'(s)d_i'(s) < d_i'(s) \tag{2.3.10}$$

if $\hat{H}$ is row dominant.

The proof of this theorem follows as a direct consequence of Ostrowski's Theorem. Consider $\bar{Z} = \hat{H} = \bar{F} + \hat{Q}$, which makes $\hat{z}_{ii}^{-1} = h_{ii}^{-1}$ and $z_{ii} = \hat{h}_{ii} = f_i + \hat{q}_{ii}$. Then

$$\phi_i(s_o) = \max_{\substack{j \\ j \neq i}} \frac{d_j(s_o)}{|f_j + \hat{q}_{jj}(s_o)|} \tag{2.3.11}$$

if $\hat{H}$ is column dominant, or

$$\phi_i'(s_o) = \max_{\substack{j \\ j \neq i}} \frac{d_j'(s_o)}{|f_j + \hat{q}_{jj}(s_o)|} \tag{2.3.12}$$

if $\hat{H}$ is row dominant, and the desired result is obtained. The set of circles defined by (2.3.11) and (2.3.12) lie inside the corresponding Gershgorin bands. They are called Ostrowski bands in honor of the mathematician on whose theorem they are based. Theorem 2 has the following graphical consequences. Pick a particular input-output relationship characterized by a diagonal element in $\hat{H}$, say $\hat{h}_{jj}$. If the gains

$$\{f_1, f_2, \ldots, f_{j-1}, f_{j+1}, \ldots, f_k\} \tag{2.3.13}$$

are kept constant while allowing f_j to vary, then one has essentially a single input single output situation. It is exactly this cause and effect relationship for which we wish to design a single loop controller. Then Theorem 2 tells us that the inverse transfer function h_{ii}^{-1} lies somewhere within the set of Ostrowski bands based on the constant gains (2.3.13). Furthermore, one may vary f_j and thus design for the appropriate stability in terms of gain margins, phase margins, other Nyquist diagram type

specifications [2, pp. 33-35] and their corresponding time domain interpretations, from the inverse Nyquist diagram with the appropriate Ostrowski bands. When these bands are small enough, the resulting composite diagram is treated as if it were just a "fuzzy" inverse Nyquist diagram

2.4. Obtaining Dominance

In Figure 2.1.2 the forward path transfer function matrix Q was forced to be square to allow the possible existence of its necessary inverse. This was done by making matrices $\bar{K}$ and $\bar{L}$ of the appropriate dimensions. It now becomes necessary to make the further assumption that the plant transfer function matrix $\bar{G}$ is square. This implies that $\bar{K}$ and $\bar{L}$ are also square or $k = \ell = m$.

As an aside, it should be mentioned that one does not wish to limit the scope of this method by assuming a plant with an equal number of inputs and outputs. Rather, at the present time there is no general way of obtaining a square $\bar{Q}$ from a nonsquare $\bar{G}$. Clearly more work is needed in this area.

When confronted with a nonsquare plant, one does have the option of reverting to the direct Nyquist array method in which analogs results as to dominance, stability and the location of the closed loop transfer function exist [2, pp. 174-179. One usually chooses not to pursue this recourse for several important reasons [2, p. 155].

1) Practical experience seems to show that there is a tendency for $\hat{G}$ to be more dominant than $\bar{G}$.

2) When working with inverse systems, the simplicity of relating the open loop system properties to those of the closed loop system adds much additional insight into a design.

3) Feedback causes the width of the Ostrowski bands to rapidly become small in the inverse system. That is to say, as feedback is increased in the other loops, we make h_i tend towards $\hat{q}_{ii}^{-1}$; the uninversed system has the opposite property and requires all the other loop gains to go to zero.

As our primary interest is in designing for adequate closed loop responses, it is clear why this limitation is accepted and why one is restricted to the inverse system representation.

2.4.1. Matrix Operations

Consider the (m×m) input compensator matrix $\overline{K(s)}$. For practical reasons $\overline{K(s)}$ should have all its poles and zeros in the open left half plane. This is because it must be physically implemented by a minimum phase, asymptotically stable subsystem. It can be shown [3, p. 209] that such a matrix may be constructed from

$$\overline{K(s)} = \overline{K}_a \overline{K_b}(s) \overline{K_c}(s). \tag{2.4.1}$$

When working with inverses, this is

$$\hat{K}(s) = \hat{K}_c(s)\hat{K}_b(s)\hat{K}_a \tag{2.4.2}$$

where each of the subscripted square matrices has the following function:

1) $\hat{K}_a$ is a permutation matrix, which is to say, in each column and each row all entries are zero except one, which is unity. Its effect is

to transpose the rows of any matrix which it premultiplies. When operating on $\hat{G}$ its function is to renumber the inputs.

2) $\hat{K}_b(s)$ is constructed as the product of elementary matrices $\hat{K}_b^{(\ell)}(s)$, each of which has unity determinant and is of the following form:

$$\hat{K}_b^{(\ell)}(s) = \begin{bmatrix} 1 & 0 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \vdots & & \vdots & & \vdots & & \vdots \\ 0 & 0 & \cdots & 1 & \cdots & -z_j^{(\ell)}(s) & \cdots & 0 \\ \vdots & \vdots & & \vdots & & \vdots & & \vdots \\ 0 & 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{bmatrix} \qquad (2.4.3)$$

where each $z_{ij}^{(\ell)}(s)$ may occur in any off diagonal position and should be a rational function of s, with strictly left half plane poles and zeros. The effect of each $\hat{K}_b^{(\ell)}(s)$ is to subtract from row i a multiple $z_{ij}^{(\ell)}(s)$ of row j from the matrix which premultiplies it. When operating on the matrix $\hat{K}_a\hat{G}(s)$, the effect of the matrix $\hat{K}_b(s)$ is to recombine the parallel paths through the system such that the resulting matrix

$$\hat{K}_b(s)\hat{K}_a\hat{G}(s) = \prod_{\ell=1}^{m}(\hat{K}_b^{(\ell)}(s))\hat{K}_a\hat{G}(s) \qquad (2.4.4)$$

is diagonally dominant.

3) The matrix $\hat{K}_c(s)$ is diagonal and nonsingular. As it represents m single loop controllers, which will be designed from classical frequency loci techniques once sufficient dominance is obtained from (2.4.4), $\hat{K}_c(s)$ should contain strictly left half plane poles and zeros.

In quite an analogous fashion, one may construct the (m×m) output compensator matrix $\overline{L(s)}$ as

$$\overline{L(s)} = \overline{L_c(s)}\ \overline{L_b(s)}\ \overline{L}_a \tag{2.4.5}$$

so that

$$\hat{L}(s) = \hat{L}_a \hat{L}_b(s) \hat{L}_c(s) \tag{2.4.6}$$

where the subscripts indicate matrices whose operations correspond to those in the previous case.

Usually one designs an input compensator with $\hat{K}(s)$ as in (2.4.2) and $\hat{L} = \overline{I}_m$. It is possible to design an output compensator with $\hat{K} = \overline{I}_m$ and $\hat{L}$ as in (2.4.6). This allows some additional freedom in design as the physical situation requires. Very often the matrix $\hat{L}(s)$ will be moved around the loop and actually implemented in the input.

One may always attempt to construct a diagonally dominant $\hat{Q}$ from these elementary row or column operations. Unfortunately, this becomes quite a chore for large systems. Even with a plant of small dimension it is very possible that a satisfactory solution will be overlooked. What is needed is a more systematic method of obtaining dominance.

The first such method which has been tried is to diagonalize $\hat{G}(o)$ with a constant $\hat{k}$[6]. If $\overline{G(o)}$ is nonsingular, this procedure will yield a $\hat{Q}(s)$ which is dominant about w=o , but one would be extremely lucky if dominance is maintained throughout the D contour. Similarly, another possibility is to diagonalize $\hat{G}(s)$ as s takes on large imaginary values, but this, too, would just be a guess. Most likely dominance would fail elsewhere on the contour.

One could attempt to combine these two efforts into a more complicated expression, such as

$$\hat{K}(s) = \bar{D}_o \hat{K}_o + \hat{K}_1 s \tag{2.4.7}$$

where $\hat{K}_o$ diagonalizes $\hat{G}(o)$, $\hat{K}_1$ diagonalizes $\hat{G}(s)$ as s approaches $j\infty$, and $\bar{D}_o$ is a diagonal matrix. Unfortunately for systems of many dimensions, the necessary inversion of $\hat{K}(s)$ given by (2.4.7) becomes rather complicated, is awkward to calculate exactly, and would be difficult to implement on the actual physical plant. Some of these problems may be overcome if, instead, one used the form

$$\overline{\overline{K(s)}} = \bar{D}_o \bar{K}_o + \bar{D}_1 \bar{K}_1 / s \tag{2.4.8}$$

where $\bar{K}_o$ diagonalizes $\overline{\overline{G(s)}}$ as s approaches $j\infty$, $\bar{K}_1$ diagonalizes $\overline{\overline{G(o)}}$, and $\bar{D}_o$ and $\bar{D}_1$ are each diagonal (gain) matrices.

While either of the forms (2.4.7) or (2.4.8) may be tried, the difficulty is obtaining a dominant $\hat{Q}$ at intermediate frequencies along the Nyquist D contour remains. Furthermore, our interest is still in finding a compensating matrix which is as simple as possible. A satisfactory constant compensating matrix might exist which would suite the purpose. This leads one to investigate possible diagonalization of $\hat{G}(s)$ at $jo < s < j\infty$. Of course, the dominance of the resulting $\hat{Q}$ will have to be checked on the entire domain of s in question, but a method which gives such a possible candidate is needed.

2.4.2. Pseudodiagonalization [7]

The diagonalization of $\hat{Q}(j\omega)$ by choosing

$$\hat{K} = \overline{G(j\omega)} \tag{2.4.9}$$

would require $\hat{K}$ to be a function of s, as $\overline{G}$ is generally a complex function. One could assume $\hat{K}$ to be constant and formulate the problem as minimizing the sum of the moduli of the off diagonal terms of the resulting $\hat{Q}(s)$. Upon placing some convenient constraint on $\hat{K}$, it can be shown that the resulting minimization problem is straightforward to formulate, but impossible to solve for a strictly real matrix $\hat{K}$ [7].

This problem may be overcome if one tries instead to minimize the sum of the squares of the moduli of the off diagonal terms. While this may be done at any specific $s = j\omega$, instead the problem is generated at N discrete, independently weighted intervals of frequency ω_r. That is, by considering row j of $\hat{Q}$, the problem is to find row j of $\hat{K}$ such that the function

$$\sum_{r=1}^{N}\{\sum_{\substack{k=1\\k\neq j}}^{m}\gamma_r|\hat{q}_{jk}(j\omega_r)|^2\} \tag{2.4.10}$$

is minimized. Here each γ_r is chosen to be positive and real, and represents the weighting factor for each ω_r as $r = 1,2,\ldots,N$. While the following results are based on an input compensator and rows as this is the usual case, analogous results for output compensators or columns may be obtained in a similar manner.

Element $\hat{q}_{jk}(j\omega_r)$ for expression (2.4.10) may be written as

$$\hat{q}_{jk}(j\omega_r) = \sum_{i=1}^{m}\hat{k}_{ji}\hat{g}_{ik}(j\omega_r) \tag{2.4.11}$$

and while $\hat{g}_{ik}(j\omega_r)$ is complex, it is expressed by real and imaginary parts as

$$\hat{g}_{ik}(j\omega_r) = \alpha_{ik}^{(r)} + j\beta_{ik}^{(r)}. \tag{2.4.12}$$

Then the function (2.4.10) to minimize becomes

$$\sum_{r=1}^{N}\{\gamma_r \sum_{\substack{k=1\\k\neq j}}^{m}\Big|\sum_{i=1}^{m}\hat{k}_{ji}(\alpha_{ik}^{(r)} + j\beta_{ik}^{(r)})\Big|^2\}. \tag{2.4.13}$$

Upon introducing the convenient constraint of

$$\sum_{i=1}^{m}\hat{k}_{ji}^2 = 1 \tag{2.4.14}$$

and appending the constraint with the Lagrange multiplier λ, expression (2.4.10) requires the minimization of the function

$$\phi_j = \sum_{r=1}^{N}\gamma_r\{\sum_{\substack{k=1\\k\neq j}}^{m}\Big|\sum_{i=1}^{m}\hat{k}_{ji}(\alpha_{ik}^{(r)} + j\beta_{ik}^{(r)})\Big|^2 + \lambda[1 - \sum_{i=1}^{m}\hat{k}_{ji}^2]\}. \tag{2.4.15}$$

Taking the necessary partial derivatives with respect to $\hat{k}_{j\ell}$ and setting them equal to zero to obtain the required minimum results in

$$\frac{\partial\phi_j}{\partial\hat{k}_{j\ell}} = 0 = \sum_{r=1}^{N}\gamma_r\{\sum_{\substack{k=1\\k\neq j}}^{m}[2(\sum_{j=1}^{m}\hat{k}_{ji}\alpha_{ik}^{(r)})\alpha_{\ell k}^{(r)} + 2(\sum_{i=1}^{m}\hat{k}_{ji}\beta_{ik}^{(r)})\beta_{\ell k}^{(r)}] - 2\lambda\hat{k}_{j\ell}\}. \tag{2.4.16}$$

For simplicity the row vector $\hat{K}_j$ is defined as

$$\hat{K}_j = (\hat{k}_{j\ell}). \tag{2.4.17}$$

Similarly the matrix

$$\bar{A}_j = (a_{i\ell}^{(j)}) = \sum_{r=1}^{N}\gamma_r[\sum_{\substack{k=1\\k\neq j}}^{m}(\alpha_{ik}^{(r)}\alpha_{\ell k}^{(r)} + \beta_{ik}^{(r)}\beta_{\ell k}^{(r)})] \tag{2.4.18}$$

is introduced. $\bar{A}_j$ is symmetric and real, and at least positive semidefinite by construction, so all of its eigenvalues are real and nonnegative. This

allows (2.4.16) to be written in the form of a standard eigenvalue problem

$$\bar{A}_j \hat{K}_j^T - \lambda \hat{K}_j^T = 0. \tag{2.4.19}$$

While any eigenvector of $\bar{A}_j$ satisfies expression (2.4.19), if one constructs $\hat{K}_j$ from the smallest nonzero eigenvalue of $\bar{A}_j$, the minimum of the original function (2.4.10) is obtained.

This procedure yields a jth row of $\hat{K}$ which is of unit length and minimizes the sum of the squares of magnitude of the off diagonal terms of row j in $\hat{Q}(j\omega)$. It is clear that this must be done for all the rows $j = 1,2,\ldots,m$. Nothing prevents the use of different values of ω_r or γ_r from (2.4.10), or just a single frequency and unity weighting for all the rows. In any case, the diagonal dominance of the resulting $\hat{Q}(s)$ must always be checked at all points of interest.

The constraint (2.4.14) leads to a relatively easy computational problem. Unfortunately, while satisfying (2.4.14) it often will simultaneously lead to unacceptably small values of $\hat{q}_{jj}$. To overcome this, the alternate constraint

$$\sum_{r=1}^{N} |\hat{q}_{jj}(j\omega_r)| = 1 \tag{2.4.20}$$

may be substituted for (2.4.14). A similar analysis shows that the more difficult solution of the eigenvector problem

$$\bar{A}_j \hat{K}_j^T - \lambda \bar{E}_j \hat{K}_j^T = 0 \tag{2.4.21}$$

is required. Here $\bar{A}_j$ is as in (2.4.18), and $\bar{E}_j$ is the symmetric, positive semidefinite matrix defined by

$$\bar{E}_j = (e_{j\ell}^{(j)}) \sum_{r=1}^{N} \gamma_r [\alpha_{ij}^{(r)} \alpha_{\ell j}^{(r)} + \beta_{ij}^{(r)} \beta_{\ell j}^{(r)}]. \qquad (2.4.22)$$

The idea of pseudodiagonalization may be extended by considering compensating matrices dependent upon frequency as in (2.4.7) where higher power terms in s may be included as needed. The general comments of the previous subsection still motivate one against such a choice. The problem is further compounded in that minimization techniques as this in no way guarantees a resulting matrix $\overline{K(s)}$ which has strictly left half plane poles and zeros.

As can be seen, obtaining open loop dominance is very often a function of the physical situation at hand. The degree of dominance is usually dependent upon the extent of complexity the system designer will allow to be actually implemented. Certainly before these methods can be put into a more general use, additional work on obtaining dominance is needed.

2.4.3. Closed Loop Dominance via Dynamic $\overline{F(s)}$

One may consider obtaining dominance in the closed loop system. If $\overline{G(o)}$ is nonsingular, there always exists a constant input compensating matrix $\bar{K}$ as

$$\bar{K} = \hat{G}(o) \qquad (2.4.23)$$

because $\overline{G(o)}$ is real. This will make $\hat{Q}(o)$ diagonal and dominance will remain for at least a small region close to the origin in the complex plane. Consider

$$\overline{F(s)} = \overline{F(o)} + \overline{F_1(s)} \tag{2.4.24}$$

where $\overline{F(o)}$ is diagonal and scalar. $\overline{F_1(s)}$ has zero diagonal elements. Its off diagonal elements, which are a function of frequency, are chosen so that the resulting dominance of $\hat{H}$ in the closed loop system is improved. As $F_1(o)$ is constructed to be zero, the resulting F(s) yields transient compensation of the interaction.

The scope of the method has not yet been well defined [8]. Such a method has never actually been physically implemented. If it would be, great care in the amount of phase advance introduced and its effect on the dynamic behavior of the system should be carefully considered. The sensitivity of the system with respect to changes in the plant will also have to be investigated, as a small change in $\overline{Q}$ may severely change the degree of dominance when cancellation of terms in $\overline{F} + \hat{Q}$ are large.

2.5. Stability

In multivariable control problems, the allowable gain in one loop is very often a complicated function of the gains in the other loops. For design purposes one would like to have a good idea of what combinations of gains result in asymptotic closed loop behavior. Permissable feedback values could then be chosen, perhaps with the additional requisite that stability is maintained as these gains vary a prescribed amount from their design values. In addition, it may be desirable to design the closed loop system so that it does not become unstable in the case of some catastrophic event,

such as a transducer or actuator failure. Before showing how this is possible some results of a general nature are necessary.

2.5.1. Frequency Domain Criteria for Stability

Theorem 3 [2, p. 141]: Let $\bar{Q}$ have an inverse $\hat{Q}$ as defined by (2.2.3), which has p_o poles in the right half plane. Let $|\hat{Q}|$ map the Nyquist contour D into $\hat{\Gamma}_Q$ and $|\hat{H}|$ map D into $\hat{\Gamma}_H$. As s goes once around D, $\hat{\Gamma}_Q$ will encircle the origin $\hat{N}_Q$ times, while $\hat{\Gamma}_H$ encircles the origin $\hat{N}_H$ times, where all positive encirclements are clockwise. Then the resulting closed loop system is asymptotically stable if and only if

$$\hat{N}_Q - \hat{N}_H = p_o. \tag{2.5.1}$$

The proof of this theorem requires showing that the return difference matrix relates the open loop poles to the closed loop poles and must be motivated on a very general ground from the theory of polynomial system matrices and the "Principle of the Argument" in complex number theory. As it is not the exact result we are seeking for design purposes, its proof may be found elsewhere [2, pp. 131-141].

Theorem 4 [2, p. 143] (Dominant Encirclement Theorem): Let the (k×k) rational complex matrix $\overline{Z(s)}$ be dominant on any simple closed contour C which has on it no pole of $z_{ii}(s)$ for all $i = 1,2,\ldots,k$. Let $z_{ii}(s)$ map C into Γ_i, which encircles the origin N_i times, and $|Z(s)|$ map C into Γ_z, which encircles the origin N_z times. All clockwise encirclements are again taken to be positive. Then

$$N_z = \sum_{i=1}^{k} N_i. \tag{2.5.2}$$

The proof of the Dominant Encirclement Theorem relies on Gershgorin's Theorem and a necessary corollary. It is rather lengthy and may be found elsewhere [2, pp. 25-26].

From these two theorem statements, a number of different stability theorems may be motivated depending upon which matrices are assumed to be dominant, and whether or not one is working with inverse systems. Rather than stating all of the possibilities, one chooses instead to give a result which is the most useful one for design purposes.

Theorem 5: (Generalized Graphical Inverse Nyquist Stability Criterion)*
Let $\hat{Q}$ and $\hat{H}$ be dominant on D, and assume that $\overline{F}$ is diagonal and independent of s.

Suppose each of the Gershgorin bands based on the diagonal elements $\hat{q}_{ii}$ of $\hat{Q}$ exclude the origin and the critical point $(-f_i,0)$. Let these bands encircle the origin $\hat{N}_{qi}$ times and encircle the critical point $(-f_i,0)$ $\hat{N}_{hi}$ times. Then the resulting closed loop system is asymptotically stable if and only if

$$\sum_{i=1}^{k} \hat{N}_{qi} - \sum_{i=1}^{k} \hat{N}_{hi} = p_o \qquad (2.5.3)$$

where p_o is the same as in Theorem 3.

Proof: When $\overline{F}$ is diagonal and independent of s, exclusion of the origin by a set of bands based upon $\hat{h}_{ii} = f_i + \hat{q}_{ii}$ is the same as the exclusion of the point $(-f_i,0)$ by a set of bands based on $\hat{q}_{ii}$. While noting that the premise of the Dominant Encirclement Theorem applies and gives the total number of

*Additional theorems and a more thorough development leading up to the statement of this theorem may be found in Rosenbrock [2, pp. 143-154].

required encirclements in both the open and closed loop system, Theorem 3 is used and the required results are obtained.

This theorem represents the Generalized Inverse Nyquist Stability Criterion for multivariable systems and has the graphical design implications as stated. It should be mentioned that it is possible to exclude the origin with one set of bands, for example based on rows, while excluding the critical point $(-f_i,0)$ by a set of bands based on columns, as is otherwise implicitly obvious in the use of the word dominance. While this theorem is of most use in its present form, it sometimes is possible to make $\hat{H}$ dominant but not $\hat{Q}$. In this special case, there is the following result

Corollary 1 [2, p. 145]: Let $\hat{H}$ be dominant on D and satisfy the dominant encirclement theorem. Then the closed loop system is asymptotically stable if and only if

$$\sum_{i=1}^{k} \hat{N}_{hi} = p_Q - Z_Q - p_o \tag{2.5.4}$$

where $\hat{N}_{hi}$ and p_o are as in Theorem 5 and p_Q and Z_Q are the poles and zeros of $|\overline{Q}|$.

Sketch of Proof: (2.5.1) in Theorem 3 states that for asymptotic stability, one must have

$$\hat{N}_Q - \hat{N}_H = p_o \tag{2.5.5}$$

while the dominant encirclement theorem states

$$\hat{N}_H = \sum_{i=1}^{k} \hat{N}_{hi}. \tag{2.5.6}$$

An application of the Principle of the Argument gives

$$\hat{N}_Q = p_Q - Z_C \tag{2.5.7}$$

and combining (2.5.5) through (2.5.7) yields the desired statement. Notice also that these results can be useful if $|\bar{Q}|$ is known algebraically.

Underlying the stability theorems in either the open or closed loop case is the idea of diagonal dominance. If a system cannot be made dominant, the theorems say nothing.

2.6. A Finishing Remark

If any design method is to be of good use it must be physically practical, both in terms of implementation and measurement. In particular, the inverse Nyquist array method fails in this respect as those values of complex frequency which correspond to the semicircular arc on the Nyquist contour D have no physical significance.

There is a natural desire to be relieved from considering this portion of the Nyquist contour in a design. Not only is it impossible to perform actual measurements on the system for these values of s, dominance may fail in this region as well.

While theorems have been presented which lead one to believe this is possible, after Rosenbrock, the following is stated as an assumption [2, pp. 153-155]:

> 'Assumption: Those poles which determine stability cross the segment of the imaginary axis between $s = -j\omega_o$ and $s = j\omega_o$, where ω_o is some specified (real) frequency.'

Therefore, one can investigate stability by considering imaginary values of s from $0 \leq s \leq j\omega_1$. Here ω_1 is greater than ω_o and $\omega_1 \geq R$, as defined by

the inclusion of all right half plane poles and zeros of the transfer functions by the Nyquist contour D.

Being able to exclude this semicircular arc makes the practical considerations of the design method complete.

CHAPTER 3
COMPUTER PROGRAMMING

3.1. Introduction

Rosenbrock's method, as outlined in the previous chapter, represents a feasible alternative in the design of multivariable control systems. It does require a rather large amount of calculation, even for plants of low order. This could be done by hand and the results graphed accordingly, but the necessary time and labor would make such an approach extremely unappealing.

Fortunately, with the present availability of high speed, time sharing computers this computational effort can be completely eliminated. Furthermore, with the advent of graphical display terminals, these results may be immediately graphed and examined. Finally, with the development of interactive software, a suitable control scheme can be built up as insight gained into the system is quickly reapplied until an acceptable design is reached.

Such an effort has been undertaken using a Tektronix graphics terminal with supporting software on the Coordinated Science Laboratory's DEC-10 computer.

Programming in the form of twenty-one subroutines has been developed in DEC system-10 FORTRAN code [9] and may be compiled using either the F10 or the F40 FORTRAN compiler. Graphic routines have been written with the aid of the Tektronix Advanced Graphics AG-II supportive software package [10].

All the subroutines are functionally oriented and have been kept general, to be useful in the design of any control problem. The routines operate on data which is stored in a common block.

As conversational software for the analysis and design of systems has become in vogue [11], it is hoped that these subroutines will be included as an option package in an interactive software program. At that time the common data base will have to be appropriately modified to conform with the conventions already established in the particular program used.

It is possible to combine the written subroutines themselves into an interactive software program. An example of how this may be done is included with the next chapter in investigating a particular design example.

3.2. Common Block Usage

The common data area has been divided into nine separately labeled segments, each of which serves its own functional importance. For efficient computer utilization when solving a specific problem, these common areas should be dimensioned as the particular situation dictates. The routines are therefore stored in separate files without the common data base and commented as to which labeled common block area is required.

COMMON /A1/ Block

Labeled common area /A1/ stores all of the system matrices as shown in the block diagram structure included in Figure 2.1.2. Matrices

$\overline{K(s)}$ and $\overline{L(s)}$ are each broken into constant matrices, XKA and XLA, and into frequency dependent matrices, XKB and XLB, respectively.

The transfer function of the plant is stored in the four dimensional array G as follows: considering location

$$G(I,J,K,L) \tag{3.2.1}$$

defines a storage position where

I = row index of transfer function

J = column index of transfer function

K = coefficient of (K-1) power of s in G(I,J)

L = 1 for the numerator polynomial

= 2 for the denominator polynomial.

Then when dimensioning G, it is clear that I and J must be equal and correspond to the number of inputs or outputs, K must be one more than the highest power of s occurring in any entry of G, and L is 2.

The coefficients of the polynomial terms in matrices XKB and XLB are referred to in exactly the same manner and are dimensioned as needed.

Matrices XKA, XLA, and F are all square, constant, and of appropriate dimension.

COMMON /A2/ Block

Labeled common area /A2/ stores seven single variables. In order of occurrence they are:

OMEGØ - real variable denoting the starting frequency of the Nyquist path. It will almost always be defined to be zero, but is included so that a region on the jω axis not including the origin may be investigated.

OMEGND - real variable corresponding to the end frequency on the Nyquist path on the jω axis. Referring to Figure 2.1.1, it is also the length of the radius R defining the semicircular arc in the right half plane.

DTOMEG - real variable corresponding to the change in frequency from OMEGØ to OMEGND as one travels up the jω axis.

ISTART - integer variable denoting the starting location in arrays yet to be defined. While it usually is 1, it has been included so we may plot any group of points corresponding to a particular segment of the frequency loci.

IEND - integer variable; it defines an end location as ISTART defines a starting location. On the jω axis, one is considering

$$IEND = [(OMENGD-OMEGØ)/DTOMEG] + 1 \qquad (3.2.2)$$

total number of such points.

NP - integer variable denoting the number of points between each Gershgorin or Ostrowski band plotted.

MO - integer variable correspond to number of inputs or outputs. In reference to Chapter 2, it is the order of the system or k.

COMMON /A3/ Block

Labeled common area /A3/ is a scratch area used as temporary storage in many of the computational routines. It has been further divided into three separate areas.

Area /A31/. Stores two square complex matrices C1 and C2, each which should be dimensioned (MO X MO). It also has two real variable vectors S1 and S2 of length MO.

Area /A32/. Stores two real variables, three dimensional arrays XR1 and XI1, which correspond to real and imaginary parts of the Nyquist map. Each should be dimensional as (MO X MO X IEND). When referring to the point in the complex plane stored in position

$$(XR1(I,J,N),XI1(I,J,N)) \tag{3.2.3}$$

then one is considering the (I,J) input-output relationship at a frequency corresponding to the Nth point on the frequency loci. For points on the $j\omega$ axis, the frequency is

$$\omega = (N-1) * DTOMEG \tag{3.2.4}$$

as N will take on values from ISTART to IEND.

COMMON A/33/. Stores two real variables, three dimensional arrays XR2 and XI2, each dimensioned as (MO X MO X IEND). They define positions and serve a function similar to arrays XR1 and XI1 in area /A32/.

COMMON /A4/ Block

Labeled common area /A4/ stores the real and imaginary parts of the open loop inverse transfer function $\hat{Q}$. It does so in two, three dimensional real variable arrays, QR and QI respectively. Each is dimensioned as (MO X MO X IEND) and the (I,J,N) storage position contains information in the same manner as defined around (3.2.3) and (3.2.4).

COMMON /A5/ Block

Labeled common area /A5/ stores information including the value of the Gershgorin bands in the four dimensional real value array BAND. Stored in location

BAND (I,J,N,L)

is the sum of the magnitudes not including the (I,J) magnitude at frequency N of the Q values from area /A4/. The sum of the row magnitudes are stored in L = 1 and the sum of the column magnitudes are stored in L = 2. Array BAND should be dimensioned as BAND (MO,MO,IEND,2).

COMMON /A6/ Block

Labeled common area /A6/ contains one three dimensional real valued array OST, which stores the values of the Ostrowski bands. In location OST (I,N,L) is the Ostrowski band based on $\hat{h}_{II}$ at frequency N where again L = 1 for the band based on rows and L = 2 for the band based on columns. Array OST should therefore be dimensioned as OST (MO,IEND,2).

COMMON /A7/ Block

Labeled common area /A7/ contains two important real value arrays XS and XC. XS is square and dimensioned as XS(MO,MO). XC is four dimensional and dimensioned the same as G in area /A1/ as defined by (3.2.1). In the subroutines which follow, calculations are performed with matrices XS and XC which will take on values from area /A1/ and thus allow the design of either input or output compensators.

3.3. Subroutines

The subroutines have been divided into three separate sections which in some respect correspond to the operation which they perform. Each individual routine is stored separately on disk under a file name which

corresponds to the subroutine name with an extension of SAV. Common block usage other than COMMON /A2/ has been appropriately commented but not included. Subroutines have further been commented as to what interactive procedures they perform, if any, and what additional subroutines are called in that routine, if any. A listing of the subroutines and a brief description of each of their functions is included in Appendix 2.

3.3.1. Library COMP

Library COMP contains seven computational subroutines useful in performing the necessary calculations at various stages of a design.

Subroutine MAP

The subroutine MAP computes the complex value of the transfer function matrix $\overline{G(s)}$ as s takes on values around the Nyquist path as defined by the real variables in COMMON /A2/.

The calling sequence is

```
CALL    MAP (ICIRCL)
```

where

ICIRCL = Ø maps the value of $\overline{G(s)}$ for strictly imaginary values of s on the jω axis from OMEGØ to OMEGND in discrete steps of DTOMEG. The total number of these points is

$$NTIMES = ((OMEGND-OMEG\emptyset)/DTOMEG) + 1. \qquad (3.3.1)$$

ICIRCL = 1 maps the value of $\overline{G(s)}$ as s goes around the semicircle of radius OMEGND. It does this in 99 discrete steps.

ICIRCL = 2 (or any other integer) causes s to accept values along the entire Nyquist path. It will do so in

$$(NTIMES + 99 + NTIMES) \tag{3.3.2}$$

discrete steps.

The resulting value of $\overline{G(s)}$ is a complex number whose real part is stored in XR1 and whose imaginary part is stored in XI1 in COMMON /A32/, for each g_{ij} and the N such discrete steps performed.

If a pole of G is encountered anywhwere on the imaginary axis, subroutine MAP types on the screen the message

A POLE AT OMEGA = XXXXX.XX FOR G(X,X,XXX).

It then adjusts the value of the complex frequency variable s accordingly to

$$s = (OMEGA, -DTOMEG) \tag{3.3.3}$$

so that these poles are included inside the contour D.

Subroutine MAG

The subroutine MAG takes the complex value of the functions implicitly defined in array XC and stores them in the labeled common area /A32/ in arrays XR1 and XI1. It does so at discrete frequencies defined by (3.2.4) where

$$N = 1,2,\ldots,IEND. \tag{3.3.4}$$

This subroutine is used in the design of frequency dependent compensators. The appropriate array from COMMON /A1/ is loaded into XC and MAG computes its complex value. Note that if one were to set XC = G, then MAG performs the same function as calling MAP(Ø).

Singularities of functions in XC have been programmed around in the same fashion as in routine MAP. The user is notified of these as the statement

A POLE AT OMEGA = XXXXX.XX FOR FUNCTION (X,X,XXX)

is typed on the terminal screen.

The calling sequence is

CALL MAG

and has no calling arguments.

Subroutine MIMP

The subroutine MIMP performs the matrix multiplication of constant and matrix XS from COMMON /A7/ with the complex matrix represented by (XR1,XI1) from COMMON /A32/. It does so for the N positions defined by (3.3.4). The resulting complex matrix is appropriately stored in labeled common /A4/ area by its real part in array QR and its imaginary part in array QI.

Routine MIMP finds its usefulness in the design of constant matrix compensators. It also presents an easy means of transferring data from labeled common area /A32/ to labeled common area /A4/ when matrix XS is set equal to the identity matrix $\tilde{I}_m$. As all the graphics routines plot frequency loci from data contained in arrays QR and QI from common /A4/, this is a necessary transfer to perform in persuing the map of $\overline{G(s)}$.

The calling sequence is

CALL MIMP

where no calling arguments are needed.

Subroutine MCMP

The subroutine MCMP is similar to routine MIMP except that it performs the multiplication of two complex matrices. The routine takes the product of the complex matrix defined by XR1 and XI1 from COMMON /A32/ with the complex matrix defined by XR2 and XI2 from COMMON /A33/. It then stores the resulting matrix, again by real and imaginary parts in QR and QI in COMMON /A4/. This is performed for all N as in (3.3.4).

The routine is useful in the design of frequency dependent compensators. The calling sequence, which has no arguments, is

```
CALL  MCMP
```

Subroutine GERSH

The subroutine GERSH computes the moduli of the Q values from COMMON /A4/ and stores the resulting real number in array BAND in COMMON /A5/ as

$$\text{BAND(I,J,N,1)} = \sum_{\substack{k=1 \\ k\neq J}}^{MO} \left| (\text{QR(I,K,N)},\text{QI(I,K,N)}) \right| \tag{3.3.5}$$

$$\text{BAND(I,J,N,2)} = \sum_{\substack{k=1 \\ k\neq I}}^{MO} \left| (\text{QR(K,J,N)},\text{QI(K,J,N)}) \right| . \tag{3.3.6}$$

When I = J the value stored in BAND(I,J,N,1) represents the radius d_i from expression (2.2.3) for Gershgorin bands based on the row $\hat{s}$ of $\hat{Q}$, while the value stored in BAND(I,J,N,2) represents the radius d_i' from expression (2.3.4) for Gershgorin bands based on the columns of $\hat{Q}$. This is for all N as in (3.3.4). Note that those values in BAND for $I \neq J$ are not actually Gershgorin bands but the appropriate sum of the magnitudes including

the diagonal term. This information may be useful in determining how a preliminary input compensator might be constructed in order that the diagonal dominance is achieved, and therefore has been included.

Upon executing the statement

CALL GERSH

these magnitudes are computed from QR and QI and stored in BAND.

Subroutine OSTROW

The subroutine OSTROW computes the Ostrowski bands and places the resulting value of the circular radius in array OST as discussed in the labeled common area /A6/ section. These computed values, as defined by (2.3.11) and (2.3.12), are based on the $\hat{Q}$ values from COMMON /A4/, their resulting diagonal BAND values from COMMON /A5/ and the current feedback gains in diagonal matrix XS from COMMON /A7/. In the closed loop inversed system, as one changes the amount of feedback, the size of the Ostrowski bands change. It then becomes necessary only to call subroutine OSTROW for each change of feedback, as the $\hat{Q}$ values and BAND values remain constant as the loops are closed.

The Ostrowski bands are calculated by executing the statement

CALL OSTROW.

Subroutine MMIJ

The subroutine MMIJ determines the minimum and maximum points on the composite plot of a Nyquist diagram with its Gershgorin bands. It therefore requires use of the $\hat{Q}$ values in COMMON /A4/ and the BAND values in COMMON /A5/.

The calling sequence is

```
CALL MMIJ(I,J,XMIN,XMAX,YMIN,YMAX)
```

where

I = row index of composite plot

J = column index of composite plot

XMIN = minimum horizontal value of composite plot

XMAX = maximum horizontal value of composite plot

YMIN = minimum vertical value of composite plot

YMAX = maximum vertical value of composite plot.

Subroutine MMIJ searches the composite diagram for $I = J$ and compares the MIN and MAX variables with the outer points of the composite diagram. For $I \neq J$ the same search is performed on the frequency loci without the Gershgorin bands. In either case it is necessary to initialize the MIN variables to a large positive value and the MAX values to a large negative value in calling the routine.

This subroutine serves two purposes when used in conjunction with the graphics routines. First, it determines where the screen margins should be set to when drawing a composite diagram, as the supportive software for the graphics terminal has no way of doing this when the Gershgorin bands are included. Second, it provides a means by which a common scale may be determined when plotting several such diagrams on the screen at once.

3.3.2. Library GRAPH

Library GRAPH contains six subroutines which display all the information necessary in performing a design.

Subroutine PTNY1

The subroutine PTNY1 plots the $\hat{Q}$ values in COMMON /A4/ from position ISTART to position IEND on the terminal screen. The supportive software automatically sets screen limits by the CALL CHECK statement if these have not been previously set elsewhere. The CALL DSPLAY statement causes the Nyquist plot and axis to be graphed on the screen.

The calling sequence is

CALL PTNY1(I,J)

where

I = row index of loci plotted.

J = column index of loci plotted.

Subroutine PTGR1

The subroutine PTGR1 causes the Gershgorin bands to be graphed over the Nyquist plot by use of the supportive software routine CPLOT. It plots these bands starting from position ISTART, and plots one band every NP positions, to position IEND.

The calling sequence is

CALL PTGR1(I,J,IRC)

where

I = row index of bands plotted

J = column index of bands plotted

IRC = 1 for Gershgorin bands based on rows

= 2 for Gershgorin bands based on columns.

Subroutine PTOST1

The subroutine PTOST1 plots the Ostrowski bands in exactly an analogous fashion as routine PTGR1 plots the Gershgorin bands.

The calling sequence is

CALL PTOST1(I,IRC)

where

I = diagonal position of $\hat{h}_{ii}$

IRC = 1 for Ostrowski bands based on rows

= 2 for Ostrowski bands based on columns.

Subroutine PTBIG1

The subroutine PTBIG1 interactively plots a single Nyquist plot and the corresponding Ostrowski bands using the maximum possible terminal screen size.

Upon executing the calling statement

CALL PTBIG1

the graphic display panel is erased and the message

TYPE ϕ TO RETURN, OR TYPE I,1/2 FOR H(I,I) AND OST ROWS/COLUMNS

is typed at the top of the screen. Entering ϕ again erases the screen and returns execution to the calling program. If values are entered, the screen is cleared and the supportive software is adjusted accordingly so that the desired composite diagram is plotted on the screen using routines PTNY1 and PTOST1. The same message as before is then typed at the very upper edge of the screen, allowing the process to repeat itself as often as needed.

Subroutine MAK4

The subroutine MAK4 causes the four Nyquist diagrams corresponding to the upper left (2X2) square portion of $\hat{Q}$ to be graphed on the screen. It also plots the Gershgorin bands for the two diagonal entries of the defined submatrix if desired. While the system under investigation could be constructed so that this routine would display something useful, its most meaningful application is for a two input, two output system.

The routine first searches all four of the composite diagrams for the maximum and minimum horizontal and vertical points, by four successive calls to subroutine MMIJ. This data is then used with some supportive software routines to set the scale of each plot to be equal. The Nyquist diagrams are graphed on the screen with subroutine PTNY1 and the Gershgorin bands, if desired are plotted on the diagonal loci with routine PTGR1.

The calling sequence is

CALL MAK4 (IRC)

where

IRC = 0 causes no Gershgorin bands to be plotted

= 1 causes Gershgorin bands based on the row elements in $\hat{Q}$ to be plotted

= 2 causes the Gershgorin based on the column elements in $\hat{Q}$ to be plotted.

Subroutine DRW

Subroutine DRW is the major interactive graphics routine. It basically serves a routing function and relies on the previous subroutines to actually perform a given operation. As such, it does not deal with any

physical data and does not require any common block usage. Included within subroutine DRW is entry point DRW1. A flow chart of the routine is included as Figure 3.3.1.

Upon executing the statement

CALL DRW

the message

PLOT DIAGRAM: TYPE: ∅ FOR NO, 1 FOR YES

is typed on the terminal screen. Entering ∅ causes execution to return to the calling program, while entering 1 causes execution to proceed to entry point DRW1. This entry point has been included as usually one determines whether or not a graphical output is desired from the calling program. If this is the case and it has been previously determined that a plot is desired one may instead execute the statement

CALL DRW1 .

In either case, at this point the message

ENTER -0,0- FOR 4 or -I,J- for Q(I,J)

is typed on the terminal screen. Entering nonzero integer values causes subroutine MMIJ to search for the necessary MAX and MIN values, the scaling is set and subroutine PTNY1 plots the appropriate frequency loci.

If two zeros are entered, or, after the single loci is graphed, the message

PLOT BANDS? TYPE ∅ FOR NO, 1 FOR ROWS, 2 FOR COLUMNS

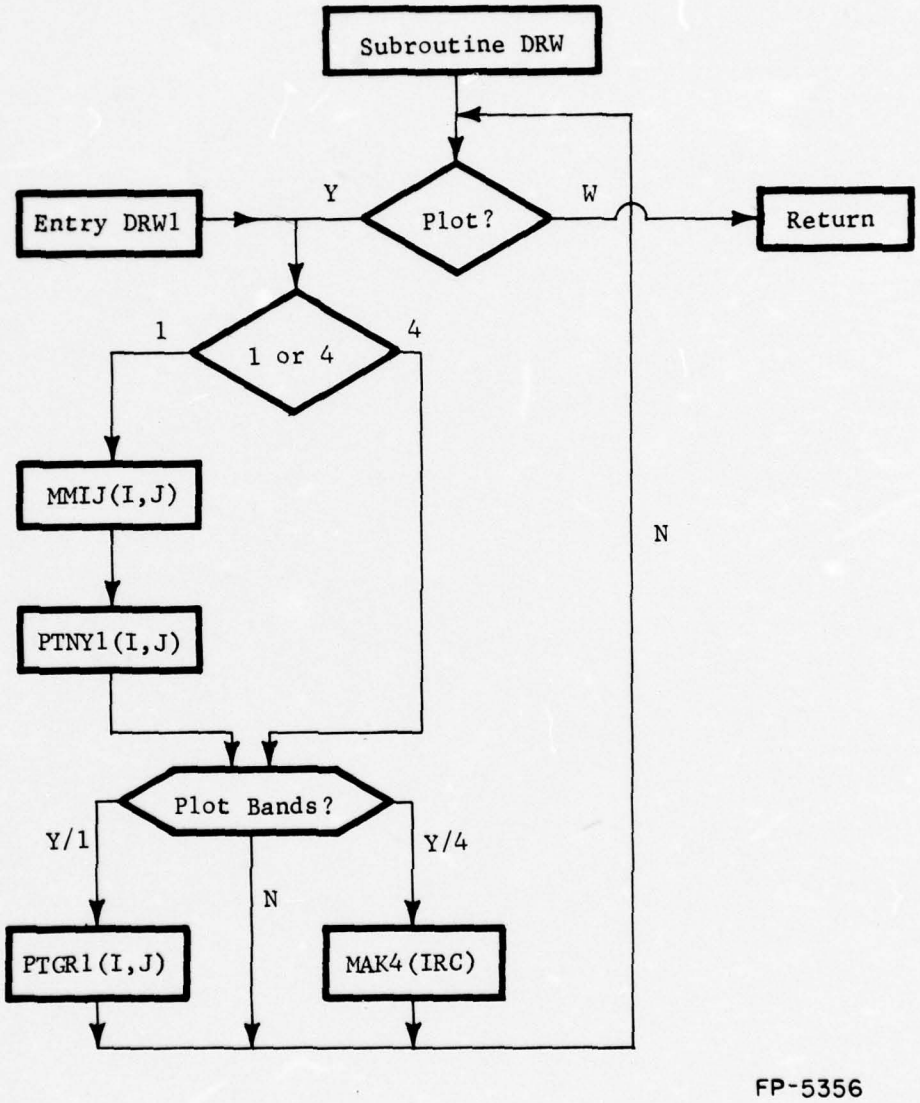


Figure 3.3.1. Flow chart for subroutine DRW.

is typed on the terminal screen. In the case of plotting four diagrams, subroutine MAK4 with the proper band identifying argument is called and the four frequency loci with the desired bands are graphed on the terminal screen. In the case of a single plot, if Gershgorin bands are to be plotted, subroutine PTGR1 with the proper band identifier is called.

In either case the routine then routes itself back to its beginning and the original message

PLOT DIAGRAMS? TYPE ϕ FOR NO, 1 FOR YES

is typed on the screen as before, allowing the process to repeat itself as many times as desired.

3.3.3. Library AUX

Library AUX contains eight subroutines of an auxiliary nature which perform the interactive modification, disk storage or outputting of various variables.

Subroutine MODIA2

The subroutine MODIA2 allows the three integer variables ISTART, IEND, and NP in labeled common area /A2/ to be modified from the computer console during a design.

Upon executing the calling statement

CALL MODIA2

the message

ENTER ISTART, IEND, NP, OR -,- AND RETURN

is typed on the screen. Entering three integer values appropriately modifies these variables, while a comma for any entry causes them to remain

the same. The routine then types the three values on the screen in a unformated output so there is no mistake as to what values they have become.

Subroutine MODXS

The subroutine MODXS performs the interactive modification or output of matrix XS in COMMON /A7/ as diagramed in Figure 3.3.2.

Upon executing the calling statement

CALL MODXS

the message

MODIFY XS? TYPE ∅ FOR NO, 1 FOR YES

is typed on the screen. Entering 1 causes the statement

TYPE I,J,X IN XS(I,J) = X

to be typed on the screen. After the values are entered, the first message

MODIFY XS? TYPE ∅ FOR NO, 1 FOR YES

again appears, allowing additional entries in XS to be changed.

Upon returning ∅ the statement

OUTPUT XS? TYPE: ∅ FOR NO, 1 FOR TTY, 2 FOR LP

is typed on the screen. After entering the appropriate value the output is performed and execution is returned to the calling program.

Subroutine MODXC

The subroutine MODXC performs the interactive modification and output of array XC as routine MODXS does for matrix XS. This again may be represented by the diagram included as Figure 3.3.2.

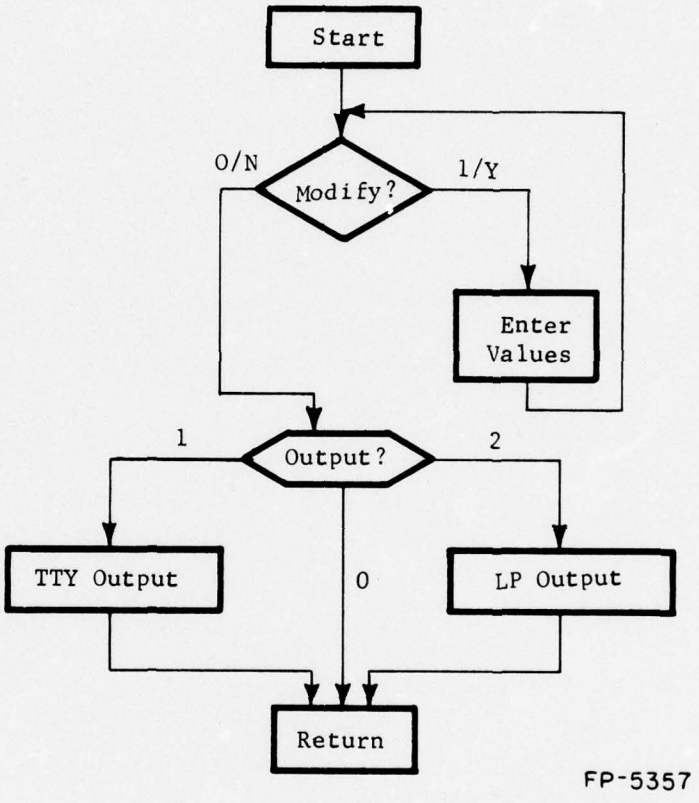


Figure 3.3.2. Functional flow diagram for subroutines MODXS and MODXC.

The calling statement is

CALL MODXC .

This causes the statement

MODIFY XC? TYPE ∅ FOR NO, 1 FOR YES

to be typed on the terminal screen. When returning 1 the statement

I,J,N,X,1/2 IN XC(I,J) = X*(S)**N,NUM/DOM

is typed on the screen and after the values are entered, the original message again appears, allowing possible further modification in array XC.

Entering ∅ causes the statement

OUTPUT XC? TYPE ∅ FOR NO, 1 FOR TTY, 2 FOR LP

to appear on the screen. Returning a value causes the appropriate output to be performed and execution is then returned to the calling program.

Subroutine QOUT

The subroutine QOUT causes the $\hat{Q}$ values in COMMON /A4/ from positions ISTART to IEND to be dumped for numerical investigation.

The calling sequence is

CALL QOUT(IDUM)

where

IDUM = 1 for terminal screen output

IDUM = 2 for line printer output

Subroutine QDSKWR

The subroutine QDSKWR stores all the $\hat{Q}$ values in COMMON /A4/ on disk file Q1.DAT.

Upon executing the calling statement

CALL QDSKWR

the message

STORE Q VALUES? TYPE ∅ FOR NO, 1 FOR YES

is typed on the terminal screen. Returning ∅ causes a return to the calling program. Typing 1 causes the formated storage of the integer 1 followed by

I,J,N,QR(I,J,N),QI(I,J,N)

on the disk file Q1.DAT. These values may be perused after execution by outputing the file, or used for the next program run by using the subroutine QDSKRD.

Subroutine QDSKRD

The subroutine QDSKRD reads the Q values from disk. It assumes that the disk file Q1.DAT exists and looks at the value stored in the second space of its first line. If this value is ∅, a return is made to the calling program and the $\hat{Q}$ values are not read. If instead this value is 1, the $\hat{Q}$ values are read and stored in arrays QR and QI in COMMON /A4/. The routine also adjusts the variable IEND in COMMON /A2/ as the largest Nth position read into these arrays.

The calling sequence is

CALL QDSKRD(IDUM)

where

IDUM = ∅ on return indicating Q values were not read

= 1 on return indicating Q values were read and stored.

Subroutine OUT

The subroutine OUT outputs all the system matrices stored in COMMON /A1/ and all the programming variables from COMMON /A2/ to either the terminal screen or the line printer. It also has the ability to call subroutine QOUT for a look at the current Q values from ISTART to IEND.

Upon executing the calling statement

CALL OUT

the message

FOR OUTPUT TYPE

Ø FOR NONE, 1 FOR TTY, 2 FOR LP

appears on the terminal screen. Typing Ø causes a return to the calling program. Typing 1 or 2 causes the message

FOR SYSTEM OUTPUT TYPE Ø

FOR Q OUTPUT TYPE 1.

Entering Ø causes the system and programming variables to be dumped, whereas typing a 1 causes the subroutine QOUT to be called.

In either case, after the appropriate output is performed, the message

TYPE Ø TO CONTINUE, TYPE 1,TO DO AGAIN

to appear on the terminal screen. Entering Ø causes execution to return to the calling program, while entering 1 causes the original message

FOR OUTPUT TYPE

Ø FOR NONE, 1 FOR TTY, 2 FOR LP

to be typed on the screen. This allows the process to repeat itself again, if desired.

Subroutine FINISH

The subroutine FINISH is executed by the calling statement

CALL FINISH.

The message

YOU ARE DONE: BEST PRINTOUT SYSTEM!!!

is typed on the terminal screen and subroutine OUT is called so that the current system values may be saved. The screen is then erased and subroutine QDSKWR is called so that Q values may be stored for further use. Execution is then terminated.

4. DESIGN ORIENTATION AND EXAMPLE

4.1. Introduction

The subroutines introduced in the previous chapter have been combined into a main program and three design packages. These have been applied to the particular design problem of a lateral autopilot to maintain leading and roll attitude.

The programming presented here is oriented to the two input, two output autopilot plant using input compensators. Output compensators may be designed in an analogous fashion by modifying the 'work' matrices from COMMON /A7/. Particular attention should be given to the order of matrix multiplication for these types of designs. Plants of higher dimensions cause no problem either, if the common data area is appropriately constructed as was discussed in Section 3.2.

4.2. Autopilot Plant

The lateral motions of an aircraft as defined by the roll, yaw and sideslip angles are coupled together, but are almost completely uncoupled to the longitudinal or pitch and plunge motions. The resulting perturbation equations to first order of the fifth order system describing these lateral motions are found [12] to be:

$$\dot{\beta} + r = \frac{Y_\beta}{mV}\beta + \phi \qquad (4.2.1)$$

$$\dot{r} + \frac{I_{xz}}{I_{zz}} \dot{p} = \frac{N_\beta}{I_{zz}} \beta + \frac{N_r}{I_{zz}} r + \frac{N_p}{I_{zz}} p + \frac{N_{\delta r}}{I_{zz}} \delta r \qquad (4.2.2)$$

$$\dot{p} + \frac{I_{xz}}{I_{xx}} \dot{r} = \frac{\ell_\beta}{I_{xx}} \beta + \frac{\ell_r}{I_{xx}} r + \frac{\ell_p}{I_{xx}} p + \frac{\ell_{\delta a}}{I_{xx}} \delta a \qquad (4.2.3)$$

$$\dot{\phi} = p \qquad (4.2.4)$$

$$\dot{\psi} = r \qquad (4.2.5)$$

State variables are defined as

ϕ = roll angle

p = roll angular velocity

ψ = yaw angle

r = yaw angular velocity

β = sideslip angle

and inputs are

δr = rudder deflection

δa = aileron deflection

where (4.2.2) and (4.2.3) must be solved simultaneously for a proper state space description. Doing so and using typical values of the coefficients for an aircraft weighing 100,000 pounds flying at 500 miles per hour at 30,000 feet [12], the resulting state space representation of the plant is

$$\begin{bmatrix} \dot{\beta} \\ \dot{r} \\ \dot{p} \\ \dot{\phi} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} -.0297 & -1.0 & 0 & .0438 & 0 \\ .331 & -.0042 & -.046 & 0 & 0 \\ -1.135 & .238 & -.795 & 0 & 0 \\ 0 & 0 & 1.0 & 0 & 0 \\ 0 & 1.0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \beta \\ r \\ p \\ \phi \\ \psi \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ -.38 & .067 \\ -.04 & 1.587 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \delta r \\ \delta a \end{bmatrix} \quad (4.2.6)$$

Computer analysis [11] shows the system to be controllable and have eigenvalues of

$$\begin{aligned} \lambda_1 &= 0.0 \\ \lambda_2 &= -.886 \\ \lambda_3 &= +.0045 \\ \lambda_{4,5} &= +.026 \pm j\ .644 \end{aligned} \quad (4.2.7)$$

indicating an open-loop-unstable system.

Before a design can begin, a suitable choice of output variables must be made. Intuitively one would control the rudder deflection with the yaw angle and the aileron deflection with the roll angle. Investigating the original regulator design made from minimizing a performance index [12] shows this primarily to be the case, where the respective angular velocities also show high gains in the corresponding feedback path.

In light of this information, the angles ψ and ϕ are chosen as the outputs of the system. These angular velocities could instead have

been chosen, but the all integrator block diagram of the plant (Figure 4.2.1) shows this would result in some unobservable states and would regulate the angular velocities and not the angular positions of the aircraft.

Computer analysis [11] shows the system to be observable and routinely yields a transfer function description of the plant as

$$\begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix} = \frac{1}{DET} \begin{bmatrix} -[.3807\ s^3+.3121\ s^2+.00893\ s+.0195] & -s[.0404\ s^2+.0503\ s+.447] \\ [.06715\ s^3-.0178\ s^2-.000588\ s+.0226] & s[1.587\ s^2+.0624\ s+.602] \end{bmatrix} \times \begin{bmatrix} \delta r \\ \delta a \end{bmatrix} \tag{4.2.8}$$

where DET = $s[s^4 + .829\ s^3+.364\ s^2+.365\ s -.00166]$ (4.2.9)

4.3. Autopilot Main Program

Before the main program for the autopilot design may be run, two files must be created on disk. The first is file Q1.DAT, which is used to store the Q values from COMMON /A4/, if desired. The integer ϕ is stored in the second space of the first line, and the file is left with no line numbers. This procedure is illustrated in Appendix 1.

The second file which must be created is the data file DATA12.FOR. This contains the data subroutine DATA12, which initializes the system data

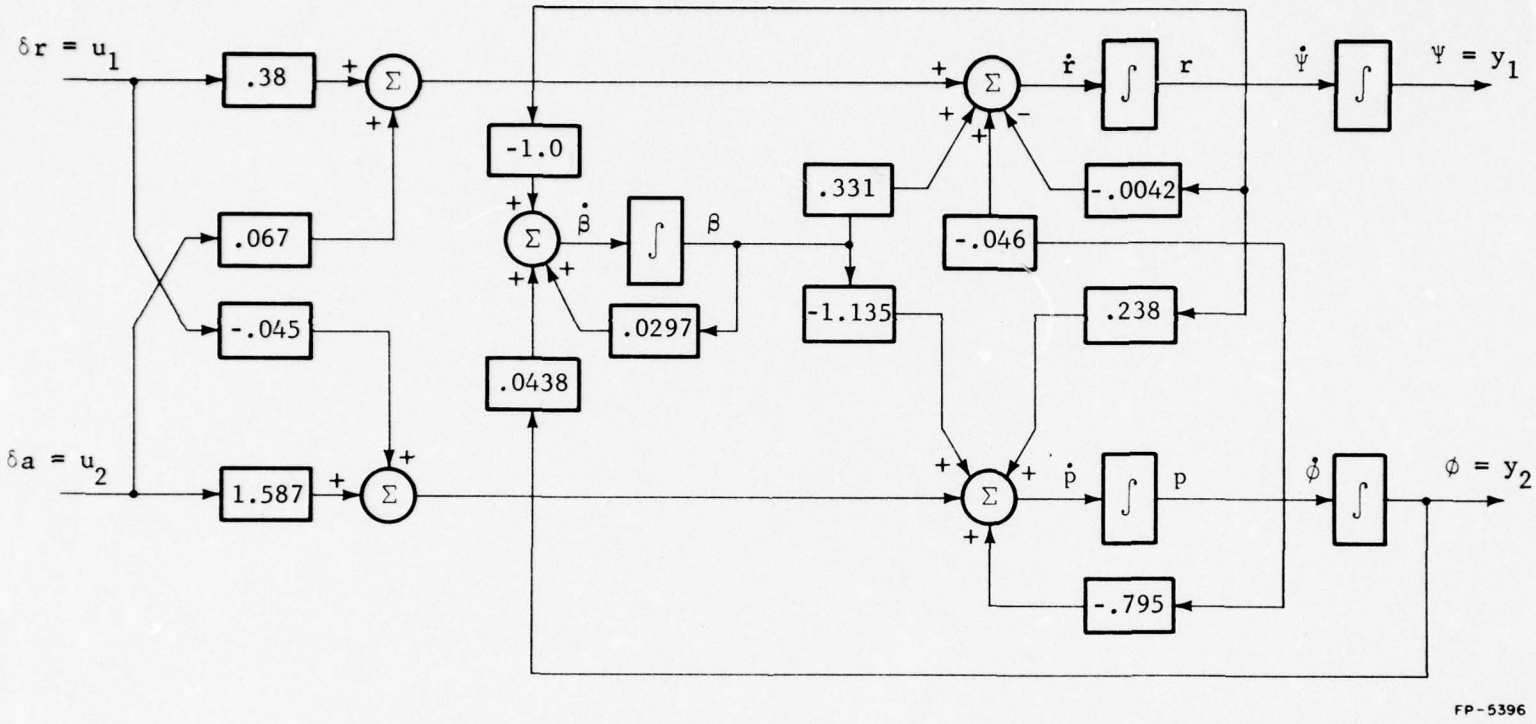


Figure 4.2.1. All integrator block diagram showing state space description of autopilot plant with BLTPLTS.

in COMMON /A1/ and the programming variables in COMMON /A2/ via standard FORTRAN assignment statements. A copy of DATA12.FOR is included in Appendix 2, along with all the autopilot programming routines.

The main program is executed from the command file MAIN.CMD as illustrated in Appendix 1. Upon its execution, the graphics terminal and supportive software are initialized, IEND is calculated, and subroutine DATA12 is called for the necessary data.

The subroutine QDSKRD is then called. If disk file Q1.DAT has $\hat{Q}$ values stored in it from the previous run, these values are automatically reloaded into arrays QR and QI in COMMON /A4/ and the routine returns the integer value IDUM = 1. This causes the message

Q VALUES WERE OBTAINED FROM DISK

to be typed on the terminal screen and execution proceeds to one of the six main program options.

If no Q values are stored from the previous run, QDSKRD returns the integer value IDUM = ϕ. This causes matrix XS to be set equal to the identity matrix, subroutine MAP ($\emptyset$) is called, causing the value of $\overline{G(s)}$ along the jw axis to be stored in COMMON /A32/. Subroutine MIMP is called, taking these values and placing them in COMMON /A4/ also.

The subroutine QINV is then called. This subroutine takes the inverse of the (2x2) square complex matrices, IEND in number, stored in COMMON /A4/. It returns the inversed values again to COMMON /A4/. Note that one could have started with the inversed system data in routine DATA12.

Taking the point by point inverse with QINV saves one the labor of computing the inverse transfer function matrix by hand. Subroutine QINV is stored on disk file QINV2.FOR and is included in Appendix 2.

In either case, COMMON /A4/ now stores the open loop system frequency data. The message

TYPE: 0 TO STOP

TYPE: 1 FOR OPEN LOOP PACKAGE

TYPE: 2 FOR INPUT COMPENSATOR PACKAGE

TYPE: 3 FOR CLOSED LOOP PACKAGE

TYPE: 4 FOR OUTPUT

TYPE: 5 TO MODIFY INTEGER COMMON /A2/ VALUES

is typed on the screen. Entering the desired integer causes execution to proceed to a particular conversational routine. Returning from one of these routines causes this same message to appear and allows the user to chose a different option, until he decides to stop.

4.4. The Interactive Design Packages

Entering ϕ to stop causes subroutine FINISH to be called. Similarly entering 4 calls routine OUT and entering 5 calls routine MODIA2. These subroutines are included with the other five auxiliary type subroutines in the file AUX FOR.

In this same manner the six computationed routines have been copies from their respective .SAV file into the file COMP.FOR. The seven

graphics routines are similarly stored in GRAPH FOR. The necessary common block usage, dimensioned for the autopilot example, is included in each subroutine of these three files.

4.4.1. Subroutine OLISP

Typing 1 for the open loop package calls the subroutine OLISP. This routine is useful in investigating the diagonal dominance of the plant and in the design of constant input compensators. It can also be used to show the effect of actuator failure at an input to the plant.

The routine has a functional flow as diagrammed in Figure 4.4.1. Upon entering the routine, work matrix XS is set equal to input compensating matrix XKA. The open loop system Q variables are stored in arrays XR1 and XI1 in COMMON /A32/ and subroutine GERSH computes the Gershgorin bands for those values. The message

DEALING W/ OPEN LOOP INPUT CONST. COMP

is typed on the screen to remind the user of the action being taken. The routing message

TYPE ∅ TO RETURN, 1 TO PLOT, 2 TO MODIFY

then appears on the terminal screen. Entering 2 calls subroutine MODXS for modification and/or intermediate output of the constant input compensator and upon its return the same directioning message is typed on the screen. Entering ∅ to return sets system matrix XKA equal to the routine's work matrix XS, and causes the execution to continue at the main program.

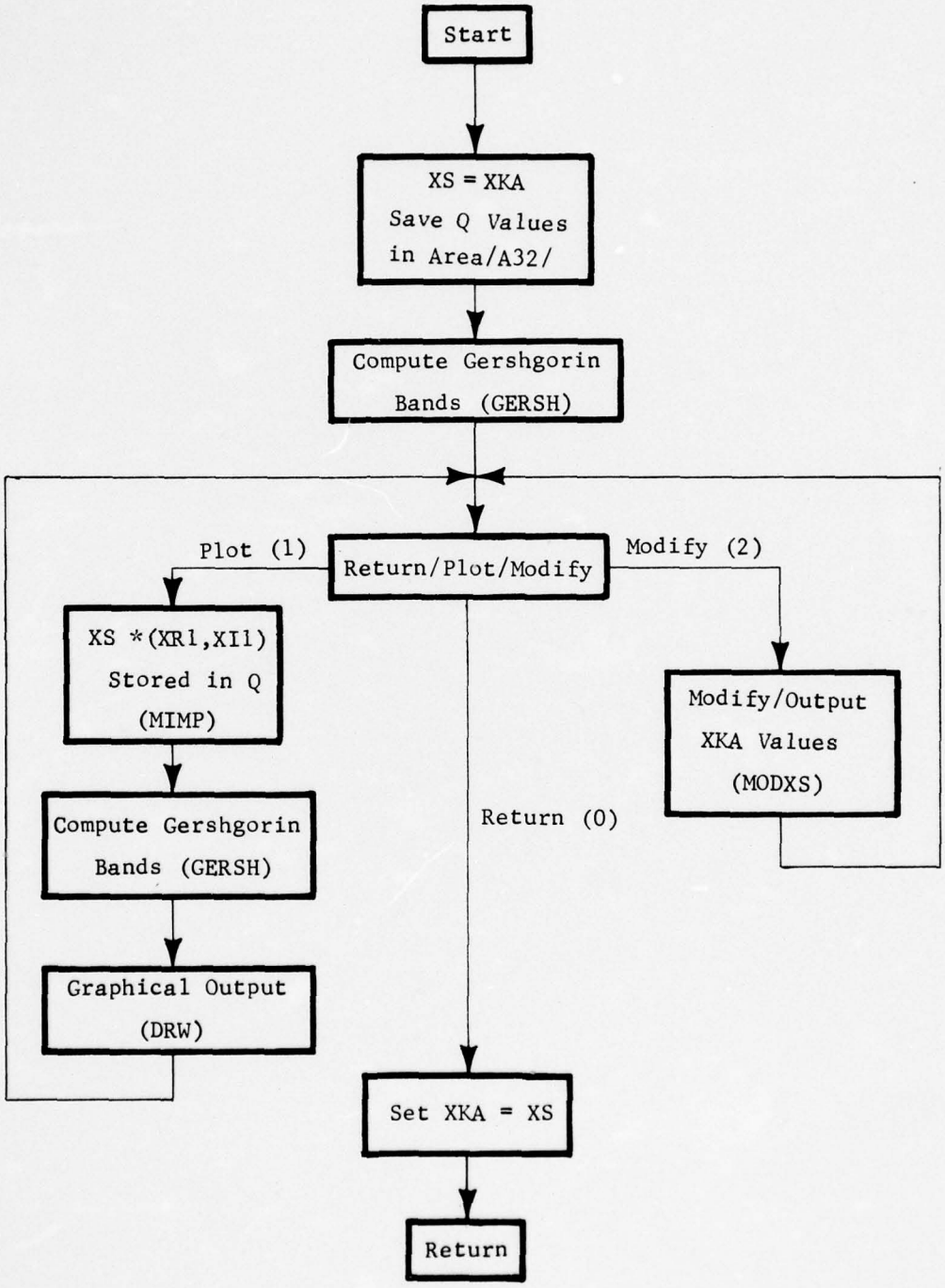


Figure 4.4.1. Functional flow chart of subroutine OLISP.

If instead one chooses to display a graphical output by entering 1, the following happens. Subroutine MIMP takes the matrix product of matrix XS and the original $\hat{Q}$, values stored in COMMON /A32/ and places the result in arrays QR and QI in COMMON /A4/. Subroutine GERSH computes the moduli of the new $\hat{Q}$ values and subroutine DRW is called at entry point DRW1 for a graphical display as described in Section 3.3.2. After all the desired plots are inspected, the routing message

TYPE: ϕ TO RETURN, 1 TO PLOT, 2 TO MODIFY

again appears on the screen, allowing further modification and plotting, or a return to the main program.

When XS is set equal to the identity matrix, the information displayed is the open loop plant. Otherwise it is the compensated plant.

The important thing to note is that when returning to the main program, the values of $\hat{Q}$ are those of the last input compensator used. If it is not desired to change these values from what they were when the package was entered, matrix XS should be set equal to the identity matrix and the results plotted once. The effect of successive changes in XS are not accumulative unless a return is made to the main program.

One may exploit this particular quirk as follows. Let the plant $\hat{Q}$ values be stored on disk. This is usually the case at this stage in a design. Enter this design package and set matrix XS equal to the identity matrix, except for one diagonal entry. This is set to zero to represent an actuator failure at that input. Return to the main program, then return to this open loop design package and set XS equal to the desired design value if the input compensator. The resulting frequency loci will be those when this actuator has failed. When compensator XKA is not diagonal, this

type of failure may have astounding effects on the resulting stability region of the closed loop system. Note that the $\hat{Q}$ values are partially destroyed by this type of procedure, and therefore should be saved beforehand.

4.4.2. Subroutine IFDCD

Typing 2 for the input compensator package option in the main program calls subroutine IFDCD. This routine is very similar to the open loop package OLISP. The difference between the two is that here the input compensator is allowed to be a function of frequency. It therefore finds its usefulness in those types of designs, but it may be used for investigation of the plant frequency loci or show the effect of actuator failures as in subroutine OLISP. A functional flow chart of routine IFDCD is included as Figure 4.4.2.

Upon entering this package, the systems $\hat{Q}$ values from COMMON /A4/ are stored in arrays XR2 and XI2 in labeled common area /A33/. The work matrix XC is set equal to the input frequency dependent matrix XKB and subroutine GERSH is called to compute the sum of the moduli of the Q values. The message

DEALING W/ FREQ. DEPNDT. INPUT COMP.

is typed on the screen.

The user then has the same three options as in the previous routine as the routing message

TYPE: ∅ TO RETURN, 1 YO PLOT, 2 TO MODIFY

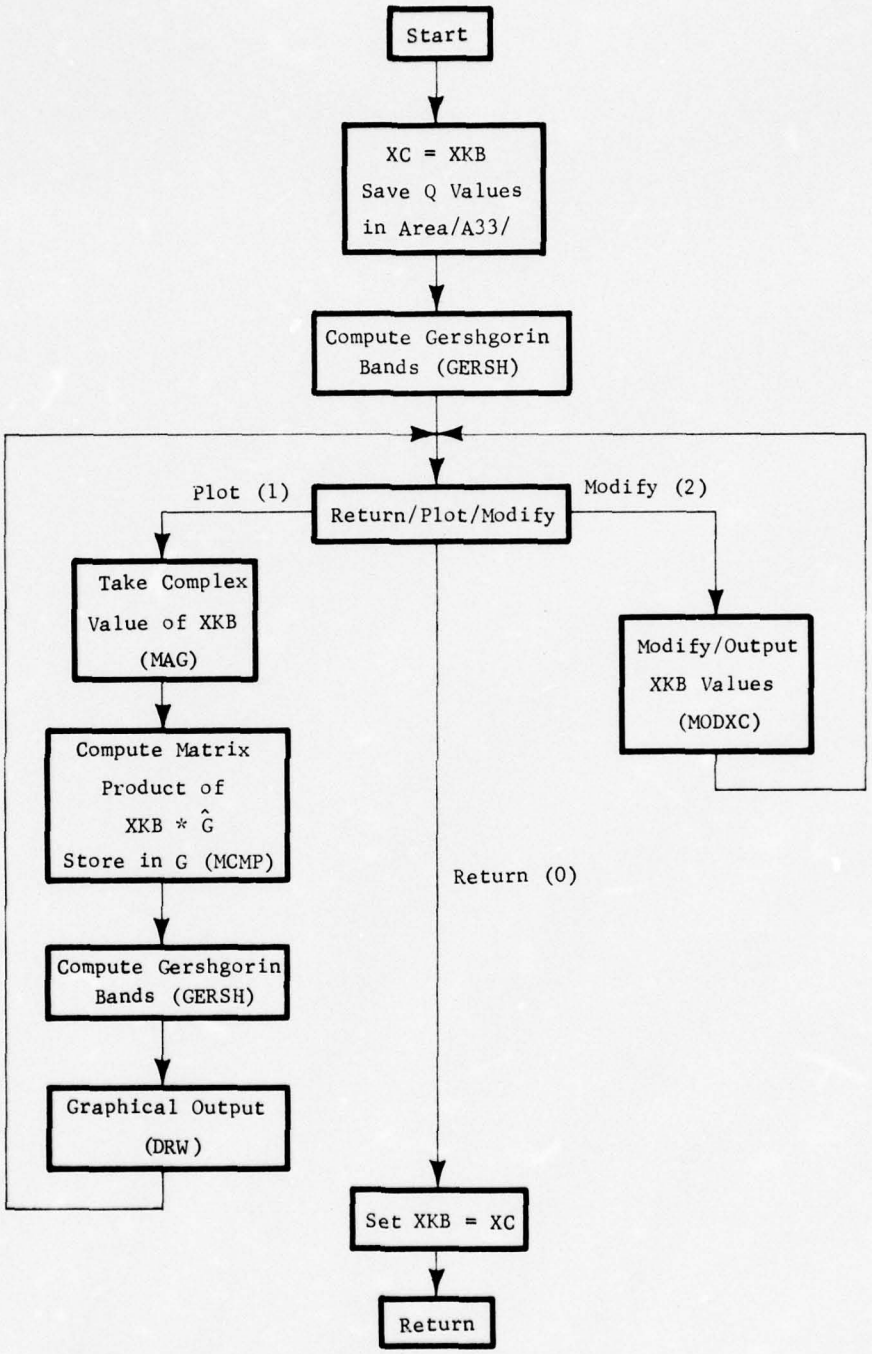


Figure 4.4.2. Functional flow chart of subroutine TFDCP.

appears on the terminal screen. Entering 2 calls subroutine MODXC to modify values in the inverse transfer function of the input compensator and/or cause its intermediate output to be performed. Upon completion, the same routing message is typed on the screen. Entering ϕ to return sets array XKB equal to the work array XC and causes a return to the main program.

Entering 1 to plot causes the following to happen. Subroutine MAG takes the complex value of the input compensator XKB stored in array XC and subroutine MCMP performs the product of these complex values with those values in $\hat{Q}$ at the entrance of the routine, stored in COMMON /A33/. This matrix product is stored in the $\hat{Q}$ arrays in COMMON /A4/. Subroutine GERSH then computes the moduli of the compensated plant and subroutine DRW is called at entry point DRW1 for a graphical display of the computed information. After all the desired plots are inspected, the original routing message appears on the screen allowing for further modification or a return to the main program.

The effects of repeative changes in array XC are not accumulative and those values of the system stored in the $\hat{Q}$ arrays in COMMON /A4/ are for the last input compensator used. Caution should again be exercised when leaving the package and entering another package from the main program.

4.4.3. Subroutine CLSP

Typing 3 for the closed loop package option in the main program calls subroutine CLSP. This programming segment allows for the interactive modification of the feedback matrix, $\bar{F}$, the computation of the Ostrowski bands, and the graphical display of the resulting closed loop composite

inverse Nyquist diagrams using the maximum possible screen size. The functional flow chart of this package is included as Figure 4.4.3.

When entering the routine, work matrix XS is set equal to the value of the feedback matrix F from COMMON /A1/. Subroutine GERSH is called to compute the Gershgorin bands and subroutine OSTROW computes the value of the Ostrowski band radii from the Gershgorin bands. The message

DEALING W/ CLOSED LOOP SYSTEM

is typed on the terminal screen to remind the user that the matrix XS has taken on the function of the feedback matrix $\bar{F}$.

The routing message

TYPE: $\emptyset$ TO RETURN, 1 TO PLOT, 2 TO MODIFY

appears on the screen. Entering $\emptyset$ sets matrix F equal to work matrix XS and a return is made to the main program. Entering 2 to modify, results in the calling of subroutine MODXS for changing values in the feedback matrix and/or their output. The return from routine MODXS causes the original routing message to again appear on the terminal screen.

If a graphical output is desired the integer 1 is entered. This causes subroutine OSTROW to compute the Ostrowski bands and subroutine PTBIG1 is called for the graphical investigation of the diagonal closed loop inverse frequency loci. Returning from the graphics routine causes the original routing message to be typed on the screen for further modification of feedback gains or a return to the main program for a different design option to be executed.

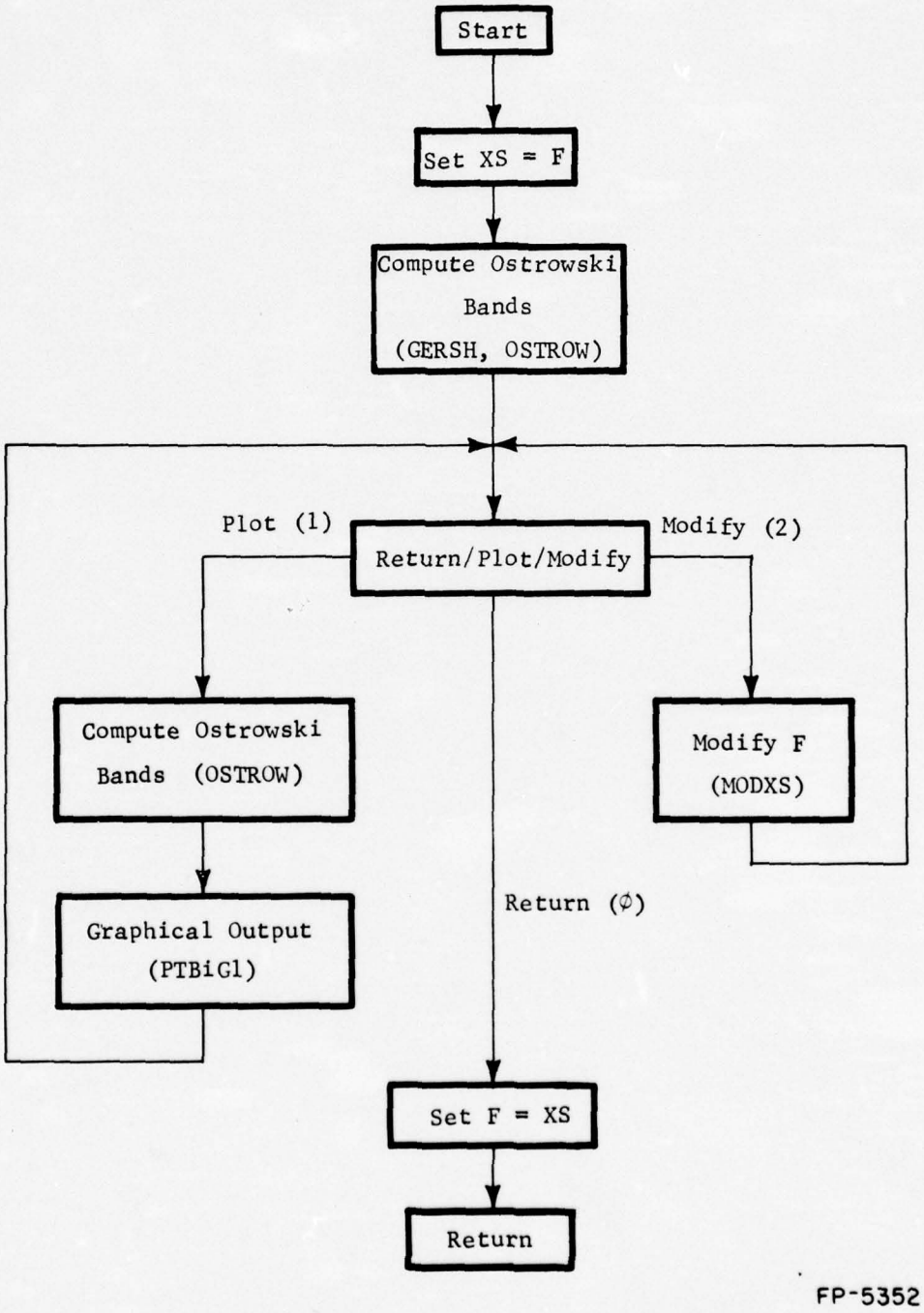


Figure 4.4.3. Functional flow chart for CLSP.

This package is also useful in showing the effects of transducer failures. By changing a specific feedback element from its design value to zero with the interactive modifying routine MODXS, the composite frequency loci of the system with this failure may be investigated.

4.5. Discussion of Autopilot Example

The autopilot plant transfer function (4.2.8) and (4.2.9) and the programming variables in COMMON /A2/ were assigned in data subroutine DATA12. Constant input compensator XKA was initialized to the identity matrix as well. The open loop package OLISP was used to investigate the composite frequency loci of the plant.

The plant without constant matrix compensation is observed to be highly nondominant (Figures 4.5.1-4.5.7). Several different constant input compensator matrices were constructed in order to achieve diagonal dominance of the open loop system. None of these attempts were successful in achieving a diagonally dominant open loop system at all frequencies.

In fact, following Hankins [13] it can be shown that the autopilot plant cannot be put into a diagonally dominant form with a constant matrix compensator.

One attempts to obtain row diagonal dominance, as this has certain advantages when matrix compensation is placed in the forward path [14] and the system was originally constructed so this would be the case. Therefore $\hat{K}$ is partitioned by rows and each of these rows is designed by considering the row dominance in that corresponding row of $\hat{Q}$. This requires $\hat{G}(j\omega)$ to

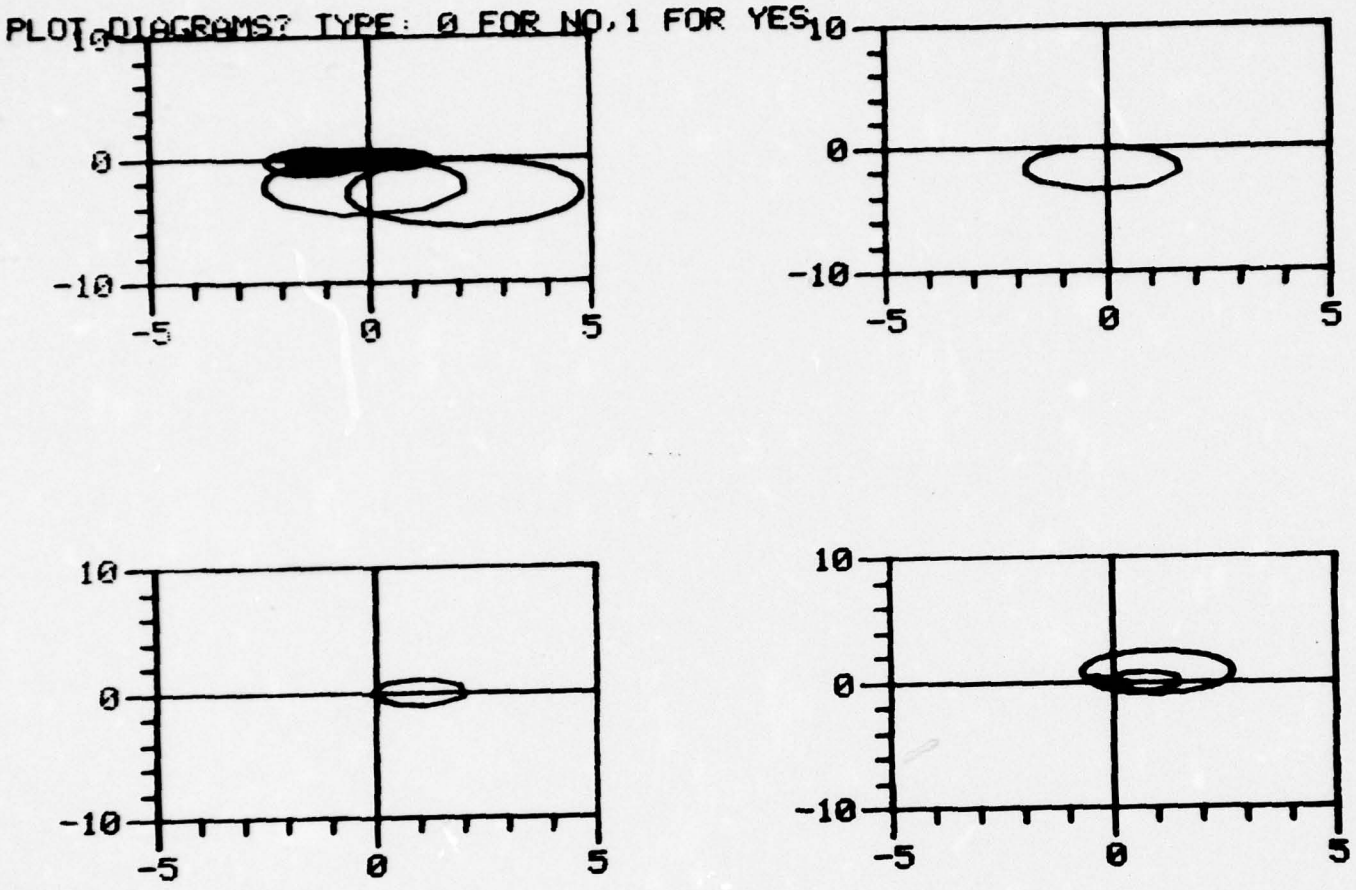


Figure 4.5.1. The four inverse Nyquist diagrams of the plant for $0 < \omega < .8$.

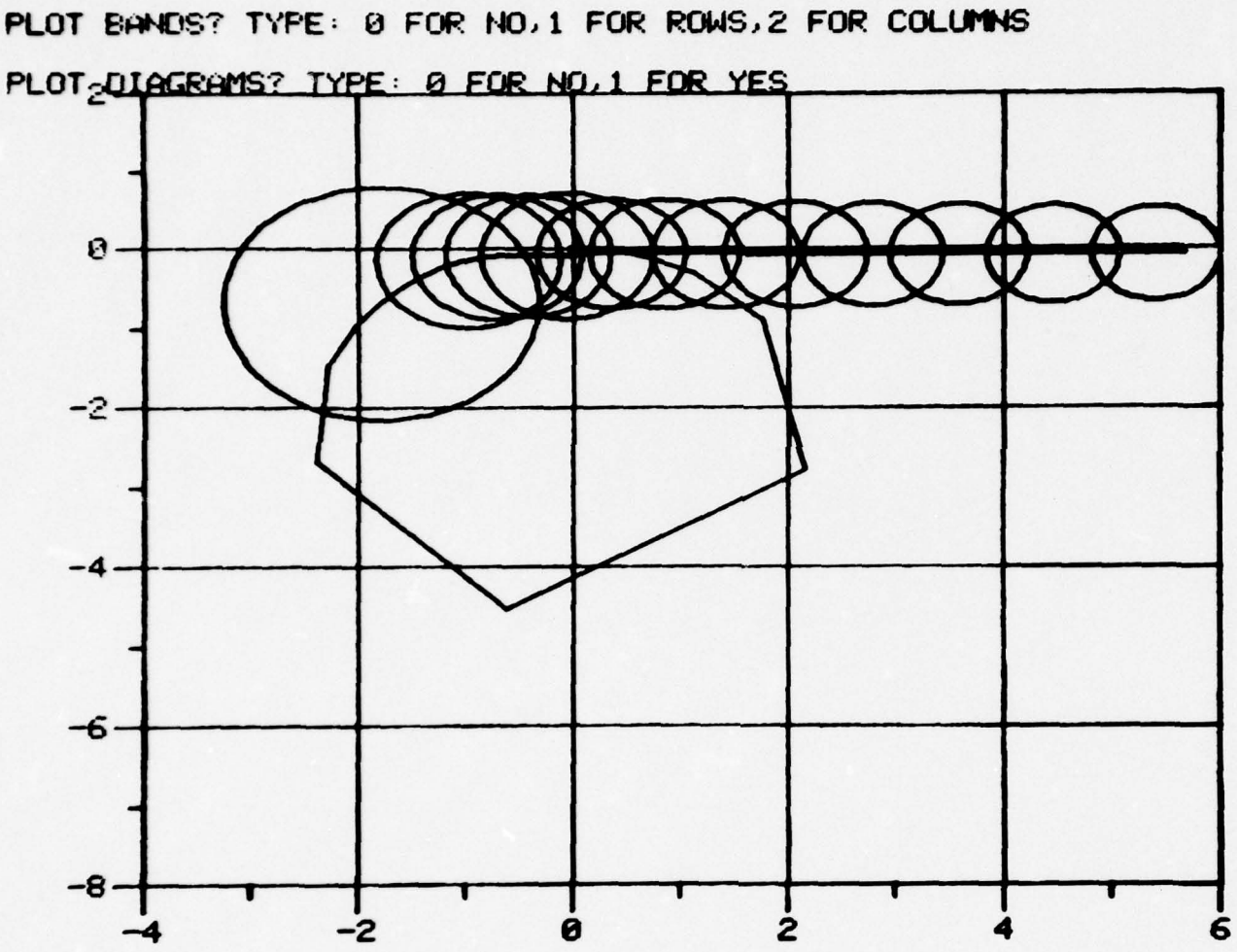


Figure 4.5.2. $\hat{g}_{11}$ with Gershgorin bands for $0 < \omega < 1.6$.

PLOT BANDS? TYPE: 0 FOR NO,1 FOR ROWS,2 FOR COLUMNS

0

PLOT DIAGRAMS? TYPE: 0 FOR NO,1 FOR YES

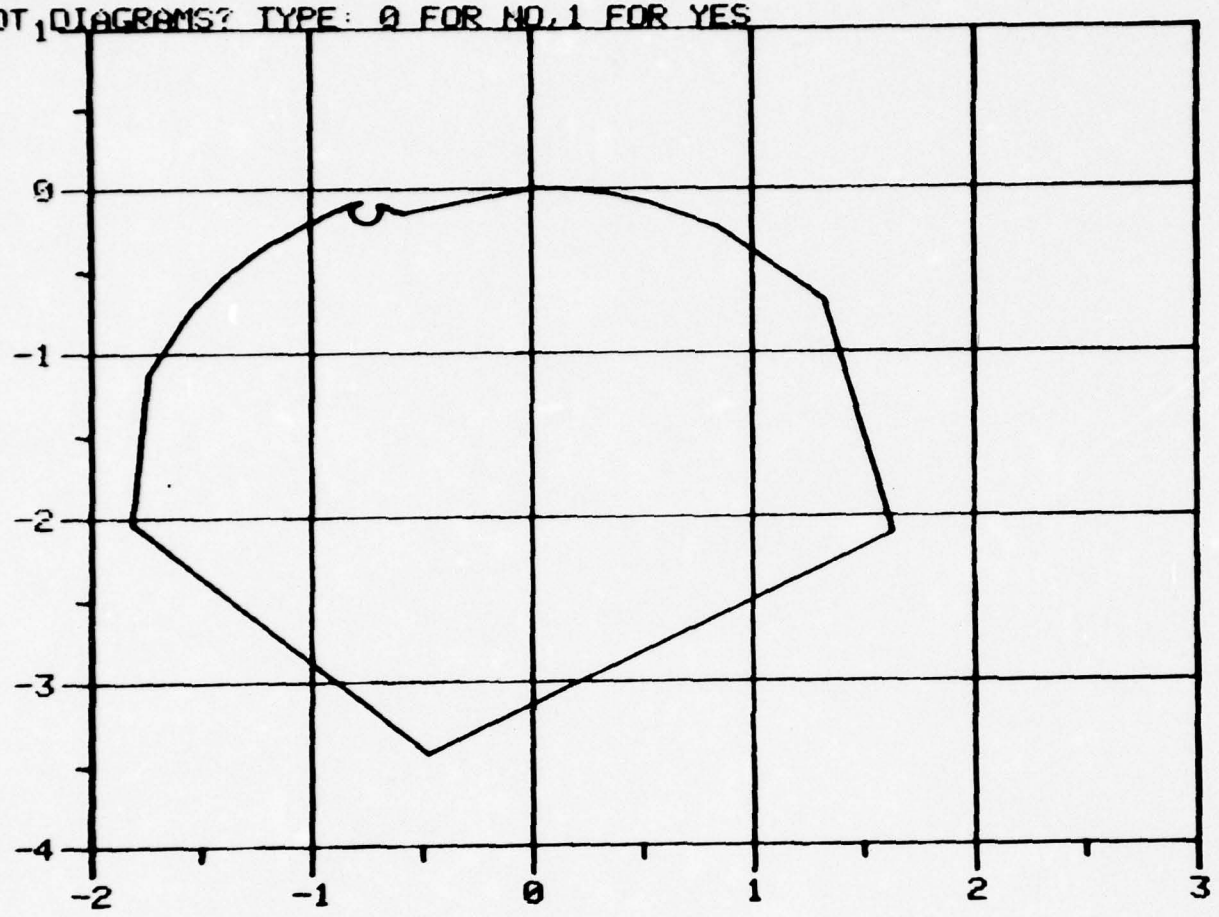


Figure 4.5.3. $\hat{g}_{12}$ for $0 < \omega < 1.6$.

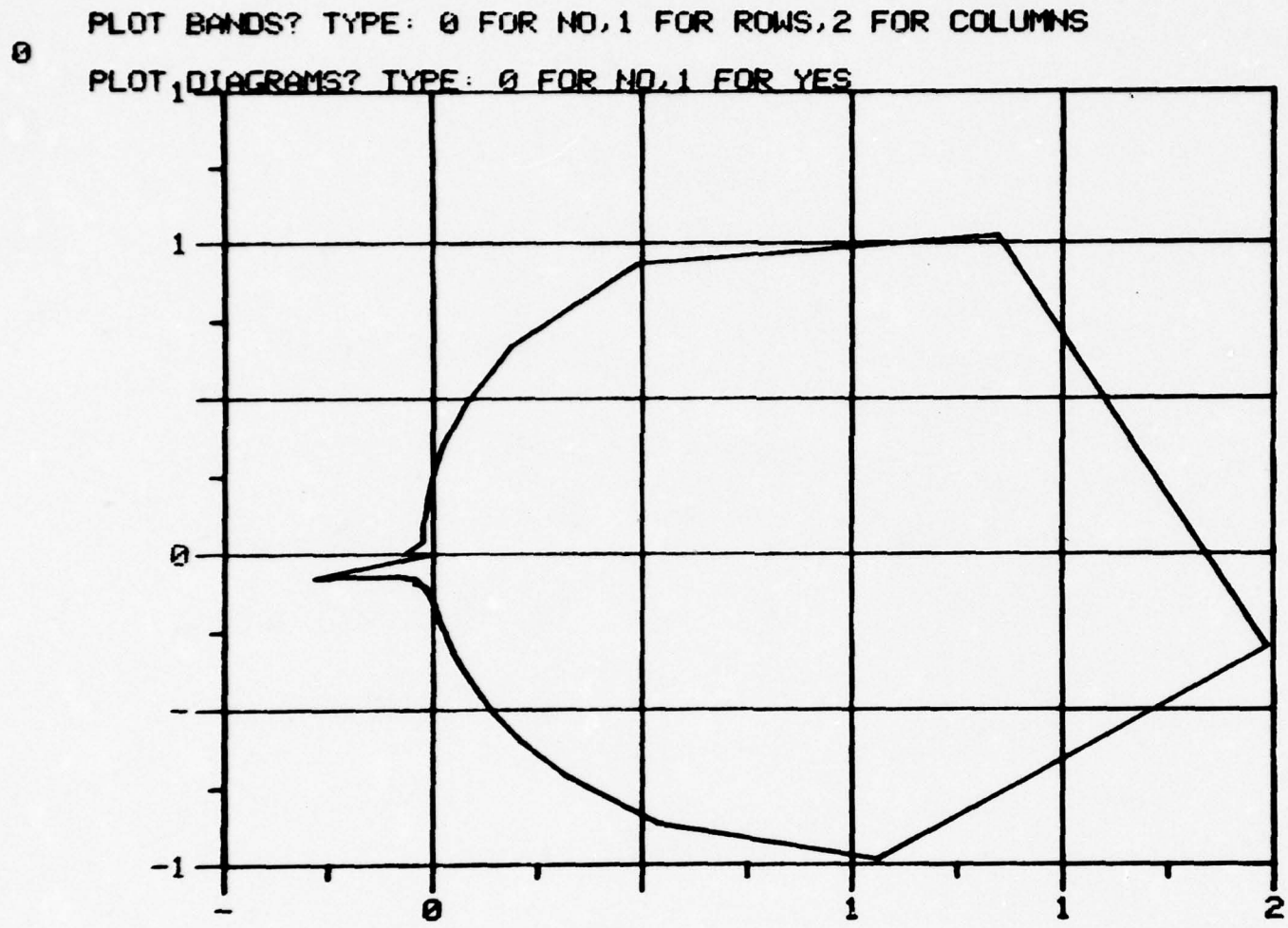


Figure 4.5.4. $\hat{g}_{21}$ for $0 < \omega < 1.6$.

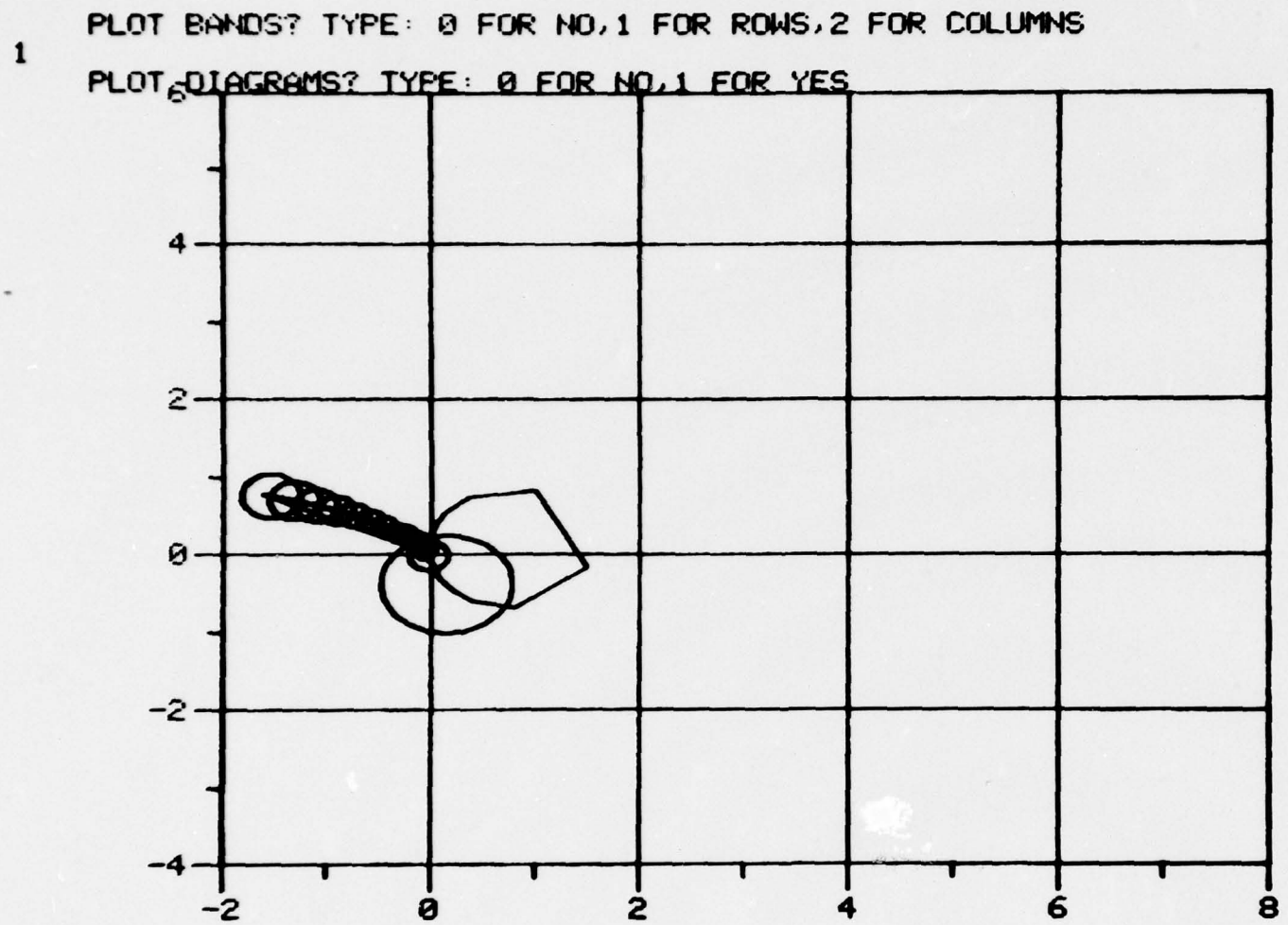


Figure 4.5.5. $\hat{g}_{22}$ with Gershgorin bands for $0 < \omega < 1.6$.

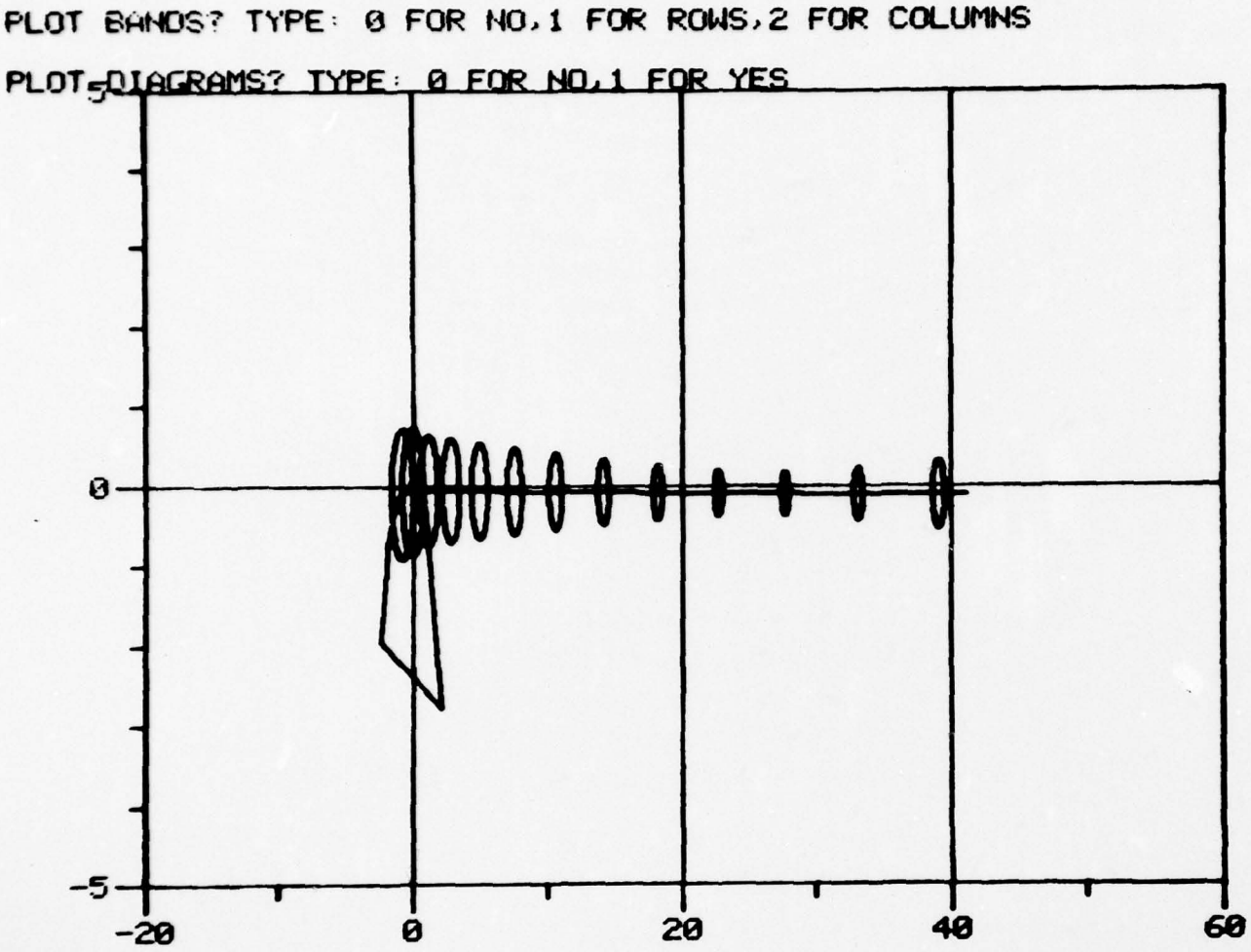


Figure 4.5.6. $\hat{g}_{11}$ with Gershgorin bands for $0 < \omega < 4.0$.

PLOT BANDS? TYPE: 0 FOR NO,1 FOR ROWS,2 FOR COLUMNS

1

PLOT DIAGRAMS? TYPE: 0 FOR NO,1 FOR YES

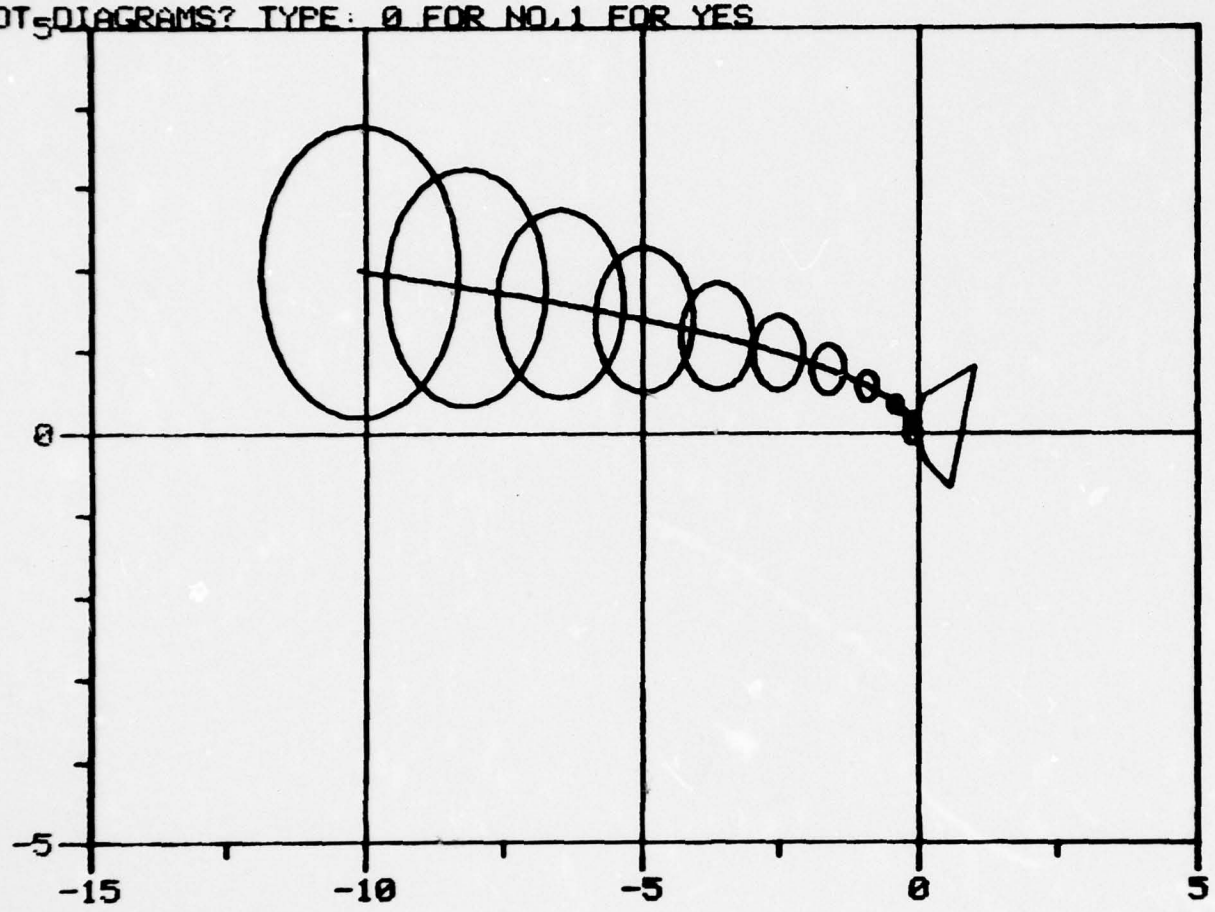


Figure 4.5.7. $\hat{g}_{22}$ with Gershgorin bands for $0 < \omega < 4.0$.

be partitioned by columns as

$$\hat{G}(j\omega) = [\alpha_1+j\beta_1 \vdots \alpha_2+j\beta_2]. \tag{4.5.1}$$

Hawkins shows that diagonal dominance cannot be obtained if the two eigenvalues of the matrix

$$\bar{C}_i = [\alpha_i\alpha_i^T+\beta_i\beta_i^T - \alpha_j\alpha_j^T - \beta_j\beta_j^T] \tag{4.5.2}$$

have the same sign. He then carries this idea further to a generalized graphical method of obtained dominance for a two input two output plant by specifying numerical bounds for the elements in such a constant compensating matrix.

For the autopilot plant, the value of the inverse transfer function at $j\omega = .1$ is

$$\hat{G}(j.1) = \begin{bmatrix} -(2.40+j1.95) & -(1.48+j1.73) \\ (.74-j.94) & (.56-j.66) \end{bmatrix}. \tag{4.4.3}$$

Although $\hat{q}_{11}$ is already both row dominant and column dominant for this frequency, C_1 is computed from (4.5.2) as

$$\bar{C}_1 = \begin{bmatrix} 3.18 & -1.05 \\ -1.05 & .608 \end{bmatrix}. \tag{4.4.4}$$

Observe that $\bar{C}_1$ is symmetric by construction and that $\bar{C}_2 = -\bar{C}_1$. The eigenvalues of $\bar{C}_1$ are found to be +3.47 and +.32. As they have the same sign, diagonal dominance in this manner can never be achieved by a constant compensator $\hat{K}$.

As open loop diagonal dominance cannot be obtained with constant compensation, a design for the autopilot regulator problem could proceed in one of the following directions:

1) Nothing prevents one from attempting to construct a nondiagonal input matrix compensator, which is a function of frequency, to obtain an open loop diagonally dominant system. However, one should keep in mind the necessary complexity of the control scheme which will result after the additional diagonal compensators for the desired input-output time domain responses are included. Present controlling methods in actual use should be surveyed to see if these means are justifiable in terms of maintenance and realiability. The resulting improvement in performance of the operating system as a function of control will ultimately be the deciding factor. Clearly more fundamental work in achieving open loop dominance via these means is needed.

2) Off diagonal elements which are a function of frequency may be inserted in the feedback path to allow for transient compensation of the interaction of the transfer functions in the closed loop system. As the inverse Nyquist method has been used only in industrial control problems which have not needed this device, all the consequences of this type of control are not fully known.

3) We chose as outputs from the state description of the plant the yaw angle and the roll angle. These motions are highly coupled through the dynamics of the aircraft with the sideslip angle. An analysis should be performed to see how the sideslip angle could be included as an output so that the resulting description of the inverse plant has a dominant form.

Unfortunately, an algorithm for such an analysis has not yet been determined.

4) It should be kept in mind that one was given highly accurate but typical values in the perturbed stated equations about an operating point for the plant description. These resulted in frequency loci which were somewhat atypical in that they exhibited peculiar behavior in certain narrow frequency regions. The validity of this precise energy description of the plant should be questioned. It of course, would be better to measure the frequency description of the actual plant in operation. This could be done at various altitudes and velocities, and under different loading conditions. Perhaps a more meaningful place to begin a design would be an investigation of the change in the input-output frequency description under such different operating conditions.

5. CONCLUDING REMARKS

The inverse Nyquist array method represents an extremely practical way in which one may solve multivariable control design problems. It is somewhat limited in that the number of system inputs and outputs must be equal. A large number of plants fit into this category, however.

Certainly an important aspect of the method is the great freedom the designer has in formulating his problem. This is not only from a standpoint of the structure of the controlling scheme. System specifications in terms of marginally acceptable performance may be included in the event of controller variations on total failure or transducer and actuator hardware. Plant variations may be dealt with in a straightforward manner.

The results presented here deal only with a linear and time invariant plant. Some theory for the nonlinear and time varying case based on describing functions does exist [2, pp. 179-184]. These results could probably be extended to sample data systems as well.

The somewhat intuitive action of the design method naturally calls for an interactive computer with graphics facility for its solution. The recent development and present availability of this hardware makes such an alternative method possible. In addition, it sheds a new light on graphically oriented design procedures in general.

This method has been applied to a variety of practical applications [2, pp. 190-219]. As it becomes more widely accepted, results will come forth allowing for additional theoretical work. In fact, very recent developments allow the diagonal dominance criterion to be considerably loosened; the need

for detailed high frequency performance in the inverse plant has been eliminated as well [14]. Without doubt, further work using this alternative type of approach will yield practical solutions to many complex design problems in the near future.

REFERENCES

[1] Horowitz, I. C. and U. Shaked, "Superiority of Transfer Function Over State Variable Methods in Linear Time-Invariant Feedback System Design," IEEE Trans. on Automatic Control, Vol. AC 20, No. 1, Feb. 1975, pp. 84-97.

[2] Rosenbrock, H. H. Computer Aided Control System Design, Academic Press, 1974.

[3] Rosenbrock, H. H. State Space and Multivariable Theory, Wiley, 1970.

[4] MacFarlane, A. G. J., "Return Difference and Return Ratio Matrices and Their Use in Analysis and Design of Multivariable Feedback Control Systems," Proc. I.E.E., Vol. 117, No. 10, 1970, pp. 2037-2049.

[5] Ostrowski, A. M., "Notes on Bounds for Determinants with Dominant Principal Diagonal," Proc. Am. Math. Soc., Vol. 3, 1952, pp. 26-30.

[6] Rosenbrock, H. H. "Design of Multivariable Control Systems Using the Inverse Nyquist Array," Proc. I.E.E., Vol. 116, No. 11, 1969, pp. 1929-1936.

[7] Hawkins, D. J., "Pseudodiogonalization and the Inverse Nyquist Array Method," Proc. I.E.E., Vol. 119, No. 3, pp. 337-342.

[8] Rosenbrock, H. H., "Progress in the Design of Multivariable Control Systems," Measurement and Control, Vol. 4, Jan. 1971, pp. 9-11.

[9] DECsystem-10 FORTRAN 10 Programmer's Reference Manual, Sixth Edition, November 1975.

[10] Advanced Graphics II, User's Manual, TEKTRONIX, Inc., July 1974.

[11] Bingulac, S. "LINSYS - Conversational Software for Analysis and Design of Linear Systems," CSL Report T-17, June 1975.

[12] Bryson, A. E., and Y.-C. Ho, Applied Optimal Control, Blaisdel, 1969.

[13] Hawkins, D. J., "Graphical Techniques for the Design of 2-Variable Control Systems," Proc. I.E.E., Vol. 19, No. 12, pp. 1740-1742.

[14] Mee, D. H., "Specifications and Criteria for Multivariable Control System Design," Proc. JACC, 1976, Paper 9.5.

APPENDIX 1

The data file Q1.DAT has a zero in the second space of the first line and may be made using the CREATE command as follows:

```
 CREATE Q1.DAT
Create: DSKC:Q1.DAT[404,443]
00100    0$
*ES
[DSKC:Q1.DAT[404,443]]

.

 TYPE Q1.DAT
 0

.
```

The main program stored in disk area [404,443] may be executated from the monitor command

EXE@MAIN.CMD

where the command file MAIN.CMD is

```
 AUTOMP.FOR
,OLISP.FOR
,IFDCP.FOR
,CLSP.FOR
,COMP.FOR
,GRAPH.FOR
,AUX.FOR
,DATA12.FOR
,QINV2.FOR
,SYS:AG210/SEA
```

This procedure creates relative binary files of 'FILENAME'.REL as the following:

EXE@MAIN.CMD

```
FORTRAN: AUTOMP
MAIN.
FORTRAN: OLISP
OLISP
FORTRAN: IFDCP
IFDCP
FORTRAN: CLSP
CLSP
FORTRAN: COMP
MAP
MAG
MIMP
MCMP
GERSH
OSTROW
MMIJ
FORTRAN: GRAPH
PTNY1
PTGR1
PTOST1
PTBIG1
MAK4
DRW
FORTRAN: AUX
MODIA2
MODXS
MODXC
QOUT
QDSKWR
QDSKRD
OUT
FINISH
FORTRAN: DATA12
DATA12
FORTRAN: QINV2
QINV
```

If these relative binary files are already created and data is stored in disk file Q1.DAT, the executation of the main program and its termination appears as

```
 EXE@MAIN.CMD
LINK:   Loading
[LNKXCT AUTOMP Execution]
Q VALUES WERE OBTAINED FROM DISK
TYPE: 0 TO STOP
TYPE: 1 FOR OPEN LOOP PACKAGE
TYPE: 2 FOR INPUT COMPENSATOR PACKAGE
TYPE: 3 FOR CLOSED LOOP PACKAGE
TYPE: 4 FOR OUTPUT
TYPE: 5 TO MODIFY INTIGER COMMON/A2/VALUES

0

YOU ARE DONE:BEST PRINT OUT SYSTEM!!!
FOR OUTPUT TYPE:
 0 FOR NONE,1 FOR TTY,2 FOR LP

0

STORE Q VALUES? TYPE 0 FOR NO, 1 FOR YES

0

5: @STOP

END OF EXECUTION
CPU TIME: 3.80  ELAPSED TIME: 1:30.37
EXIT
```

APPENDIX 2

The twenty-one subroutines are stored individually on disk in area [404,443] in file names corresponding to the subroutine name with extension of SAV. They have been left unprotected and therefore may be copies into a user's disk area by executing the monitor command

COPYXXXXXX.SAV[404,443]

where XXXXXX is the appropriate filename. One may copy all twenty one routines at once by using the option

COPY*.SAV[404,443]

In a similiar fashion the programming for the autopilot problem may be transferred to a different users area. These files include:

I. COMP.FOR which contains the computational routines:

1) Subroutine MAP, which computes points on the frequency loci for any region on the Nyquist contour D.
2) Subroutine MAG, which computes the complex value of any transfer function matrix.
3) Subroutine MIMP, which takes the product of a real matrix and a complex matrix stored by real and imaginary parts.
4) Subroutine MCMP, which takes the product of two complex matrices each stored by real and imaginary parts.
5) Subroutine GERSH, which computes various sums of the moduli of elements in the open loop plant.

6) Subroutine OSTROW, which computes the radi of the Ostrowski bands in the closed loop system.

7) Subroutine MMIJ, which searches for minimum and maximum values on a composite frequency locus.

See Table A.2.1 for a listing.

II. GRAPH.FOR which contains the graphics routines:

1) Subroutine PTNY1, which plots a single frequency loci on the screen.

2) Subroutine PTGR1, which plots the Gershgorin bands based on rows or columns on the screen.

3) Subroutine PTOST1, which plots the Ostrowski bands based on rows or columns on the screen.

4) Subroutine PTBIG1, which interactively causes a single composite Nyquist diagram with Ostrowski bands using the maximum possible screen size to be plotted.

5) Subroutine MAK4, which plots four separate frequency loci with Gershgorin bands on the screen simultaneously.

A listing of this program is given in Table A.2.2.

III. AUX.FOR which contains the auxiliary routines:

1) Subroutine MODIA2, which interactively modifies the range of the frequency loci and the number of Gershgorin or Ostrowski bands.

2) Subroutine MODXS, which interactively modifies and/or outputs scalar matricies.

3) Subroutine MODXC interactively modifies and/or outputs frequency dependent compensators.

Table A.2.1. COMP.FOR listing

```
00100   C       ***************************************************************
00200   C       ***************************************************************
00300           SUBROUTINE MAP(ICIRCL)
00400   C       ***************************************************************
00500   C       TAKES COMPLX MAP OF G INTO (XR1,XI1)
00600   C       ICIRCL = 0 FOR S BETWEEN OMEG0 AND OMEGND ONLY
00700   C              = 1 FOR S ON THE SEMICIRCULAR ARC ONLY
00800   C              = 2 OR ANY OTHER VALUE FOR ENTIRE NYQUIST PATH
00900   C       ***************************************************************
01000           IMPLICIT COMPLEX(C)
01100           COMMON/A2/OMEG0,OMEGND,DTOMEG,ISTART,IEND,NP,MO
01200   C       ***************************************************************
01300   C       USES COMMON/A1/,COMMON/A32/ ALSO
01400   C       ***************************************************************
01500           COMMON/A1/G(2,2,6,2),XKB(2,2,3,2),XLB(2,2,3,2),
01600           1 XKA(2,2),XLA(2,2),F(2,2)
01700           COMMON/A32/XR1(2,2,201),XI1(2,2,201)
01800   C       ***************************************************************
01900   970     FORMAT(X,'A POLE AT OMEGA = ',F7.2,2X,'FOR G(',I1,',',
02000         / I1,',',I3,')')
02100           NTIMES=((OMEGND-OMEG0)/DTOMEG)+1
02200           DO 780 I=1,MO
02300           DO 780 J=1,MO
02400           IF(ICIRCL.EQ.1) GO TO 751
02500   C       MAPS TOP STRIP OF JW AXIS
02600           DO 750 N=1,NTIMES
02700           IF(N.GT.IEND) GO TO 780
02800           OMEGA=OMEG0+(N-1)*DTOMEG
02900           SIGMA=0.0
03000           GO TO 730
03100   720     SIGMA=-DTOMEG
03200           TYPE 970,OMEGA,I,J,N
03300   730     CSS=CMPLX(SIGMA,OMEGA)
03400           CNUM=(0.0,0.0)
03500           CDOM=(0.0,0.0)
03600           DO 740 K=2,6
03700           CNUM=CNUM+CMPLX(G(I,J,K,1),0.0)*CSS**(K-1)
03800           CDOM=CDOM+CMPLX(G(I,J,K,2),0.0)*CSS**(K-1)
03900   740     CONTINUE
04000           CNUM=CNUM+CMPLX(G(I,J,1,1),0.0)
04100           CDOM=CDOM+CMPLX(G(I,J,1,2),0.0)
04200           IF(CDOM.EQ.(0.0,0.0)) GO TO 720
04300           C=CNUM/CDOM
04400           XR1(I,J,N)=REAL(C)
04500           XI1(I,J,N)=AIMAG(C)
04600   750     CONTINUE
04700           IF(ICIRCL.EQ.0) GO TO 780
04800   751     IF(ICIRCL.EQ.1) NTIMES=0
04900   C       MAPS SEMICIRCULAR RADIUS PART
05000           DO 760 N=1,99
05100           N1=N+NTIMES
05200           OMEGA=OMEGND*COS(N*.0314159)
05300           SIGMA=OMEGND*SIN(N*.0314159)
05400           CSS=CMPLX(SIGMA,OMEGA)
05500           CNUM=(0.0,0.0)
05600           CDOM=(0.0,0.0)
05700           DO 742 K=2,6
05800           CNUM=CNUM+CMPLX(G(I,J,K,1),0.0)*CSS**(K-1)
05900           CDOM=CDOM+CMPLX(G(I,J,K,2),0.0)*CSS**(K-1)
06000   742     CONTINUE
```

```
06100           CNUM=CNUM+CMPLX(G(I,J,1,1),0.0)
06200           CDOM=CDOM+CMPLX(G(I,J,1,2),0.0)
06300           IF(CDOM.EQ.(0.0,0.0))TYPE 970,OMEGA,I,J,N
06400           IF(CDOM.EQ.(0.0,0.0)) GO TO 760
06500           C=CNUM/CDOM
06600           XR1(I,J,N+NTIMES)=REAL(C)
06700           XI1(I,J,N+NTIMES)=AIMAG(C)
06800   760     CONTINUE
06900           IF(ICIRCL.EQ.1) GO TO 780
07000   C       MAPS BOTTOM STRIP OF JW AXIS
07100           DO 770 N=1,NTIMES
07200           N2=N+99+NTIMES
07300           OMEGA=OMEGND-(N-1)*DTOMEG
07400           SIGMA=0.0
07500           GO TO 733
07600   723     SIGMA=-DTOMEG
07700           TYPE 970,OMEGA,I,J,N2
07800   733     CSS=CMPLX(SIGMA,OMEGA)
07900           CNUM=(0.0,0.0)
08000           CDOM=(0.0,0.0)
08100           DO 743 K=2,6
08200           CNUM=CNUM+CMPLX(G(I,J,K,1),0.0)*CSS**(K-1)
08300           CDOM=CDOM+CMPLX(G(I,J,K,2),0.0)*CSS**(K-1)
08400   743     CONTINUE
08500           CNUM=CNUM+CMPLX(G(I,J,1,1),0.0)
08600           CDOM=CDOM+CMPLX(G(I,J,1,2),0.0)
08700           IF(CDOM.EQ.(0.0,0.0)) GO TO 723
08800           C=CNUM/CDOM
08900           XR1(I,J,N+99+NTIMES)=REAL(C)
09000           XI1(I,J,N+99+NTIMES)=AIMAG(C)
09100   770     CONTINUE
09200   780     CONTINUE
09300           RETURN
09400           END
09500   C       ***************************************************************
09600   C       NO INTERACTIVE USAGE
09700   C       ***************************************************************
09800   C       NO ADDITIONAL ROUTINES CALLED
09900   C       ***************************************************************
10000   C       ***************************************************************
10100   C       ***************************************************************
10200           SUBROUTINE MAG
10300   C       ***************************************************************
10400   C       TAKES COMPLEX VALUE OF XC INTO (XR1,XI1)
10500   C       ***************************************************************
10600           IMPLICIT COMPLEX(C)
10700           COMMON/A2/OMEG0,OMEGND,DTOMEG,ISTART,IEND,ID,MO
10800   C       ***************************************************************
10900   C       USES COMMON/A32/,COMMON/A7/ ALSO
11000   C       ***************************************************************
11100           COMMON/A32/XR1(2,2,201),XI1(2,2,201)
11200           COMMON/A7/XS(2,2),XC(2,2,6,2)
11300   C       ***************************************************************
11400           DO 60 I=1,MO
11500           DO 60 J=1,MO
11600           DO 50 N=1,IEND
11700           OMEGA=OMEG0+(N-1)*DTOMEG
11800           SIGMA=0.0
11900           GO TO 30
12000   20      SIGMA=-DTOMEG
```

AD-A040 762 ILLINOIS UNIV AT URBANA-CHAMPAIGN COORDINATED SCIENCE LAB F/G 9/2
FREQUENCY METHODS IN COMPUTER AIDED DESIGN OF CONTROL SYSTEMS.(U)
DEC 76 S C SPIELMAN DAAB07-72-C-0259
UNCLASSIFIED R-753 NL

2 OF 2
AD A040762

END
DATE FILMED
7-77

```
12100           TYPE 900,OMEGA,I,J,N
12200   900     FORMAT(X,'A POLE AT OMEGA = ',F7.2,2X,'FOR FUNCTION (',I1,',',
12300          / I1,',',I3,')')
12400   30      CSS=CMPLX(SIGMA,OMEGA)
12500           CNUM=(0.0,0.0)
12600           CDOM=(0.0,0.0)
12700           DO 40 K=2,6
12800           CNUM=CNUM+CMPLX(XC(I,J,K,1),0.0)*CSS**(K-1)
12900           CDOM=CDOM+CMPLX(XC(I,J,K,2),0.0)*CSS**(K-1)
13000   40      CONTINUE
13100           CNUM=CNUM+CMPLX(XC(I,J,1,1),0.0)
13200           CDOM=CDOM+CMPLX(XC(I,J,1,2),0.0)
13300           IF(CDOM.EQ.(0.0,0.0)) GO TO 20
13400           C=CNUM/CDOM
13500           XR1(I,J,N)=REAL(C)
13600           XI1(I,J,N)=AIMAG(C)
13700   50      CONTINUE
13800   60      CONTINUE
13900           RETURN
14000           END
14100   C       ***************************************************************
14200   C       NO INTERACTIVE USAGE
14300   C       ***************************************************************
14400   C       NO ADDITIONAL ROUTINES CALLED
14500   C       ***************************************************************
14600   C       ***************************************************************
14700   C       ***************************************************************
14800           SUBROUTINE MIMP
14900   C       ***************************************************************
15000   C       TAKES MATRIX MULTP. OF XS=CMPLX(XR1,XI1)
15100   C       STORES THE RESULT IN Q
15200   C       ***************************************************************
15300           IMPLICIT COMPLEX(C)
15400           COMMON/A2/D1,D2,D3,ISTART,IEND,ID,MO
15500   C       ***************************************************************
15600   C       USES COMMON/A31/,COMMON/A32/,COMMON/A4/,COMMON/A7/ ALSO
15700   C       ***************************************************************
15800           COMMON/A31/C1(2,2),C2(2,2),S1(2),S2(2)
15900           COMMON/A32/XR1(2,2,201),XI1(2,2,201)
16000           COMMON/A4/QR(2,2,201),QI(2,2,201)
16100           COMMON/A7/XS(2,2),XC(2,2,6,2)
16200   C       ***************************************************************
16300           DO 40 N=1,IEND
16400           DO 10 I=1,MO
16500           DO 10 J=1,MO
16600           C1(I,J)=CMPLX(XR1(I,J,N),XI1(I,J,N))
16700   10      CONTINUE
16800           DO 30 I=1,MO
16900           DO 30 J=1,MO
17000           CX=(0.0,0.0)
17100           DO 20 K=1,MO
17200           CX=CX+XS(I,K)*C1(K,J)
17300   20      CONTINUE
17400           QR(I,J,N)=REAL(CX)
17500           QI(I,J,N)=AIMAG(CX)
17600   30      CONTINUE
17700   40      CONTINUE
17800           RETURN
17900           END
18000   C       ***************************************************************
```

```
18100    C       NO INTERACTIVE USAGE
18200    C       ************************************************************
18300    C       NO ADDITIONAL ROUTINES CALLED
18400    C       ************************************************************
18500    C       ************************************************************
18600    C       ************************************************************
18700            SUBROUTINE MCMP
18800    C       ************************************************************
18900    C       TAKES MATRIX MULTP. OF CMPLX(XR1,XI1)*CMPLX(XR2,XI2)
19000    C       STORES RESULT IN Q
19100    C       ************************************************************
19200            IMPLICIT COMPLEX(C)
19300            COMMON/A2/D1,D2,D3,ISTART,IEND,ID,MO
19400    C       ************************************************************
19500    C       USES COMMON/A31/,COMMON/A32/,COMMON/A33/,COMMON/A4/ ALSO
19600    C       ************************************************************
19700            COMMON/A31/C1(2,2),C2(2,2),S1(2),S2(2)
19800            COMMON/A32/XR1(2,2,201),XI1(2,2,201)
19900            COMMON/A33/XR2(2,2,201),XI2(2,2,201)
20000            COMMON/A4/QR(2,2,201),QI(2,2,201)
20100    C       ************************************************************
20200            DO 50 N=1,IEND
20300            DO 20 I=1,MO
20400            DO 20 J=1,MO
20500            C1(I,J)=CMPLX(XR1(I,J,N),XI1(I,J,N))
20600            C2(I,J)=CMPLX(XR2(I,J,N),XI2(I,J,N))
20700    20      CONTINUE
20800            CX=(0.0,0.0)
20900            DO 30 K=1,MO
21000            CX=CX+C1(I,K)*C2(K,J)
21100    30      CONTINUE
21200            QR(I,J,N)=REAL(CX)
21300            QI(I,J,N)=AIMAG(CX)
21400    40      CONTINUE
21500    50      CONTINUE
21600            RETURN
21700            END
21800    C       ************************************************************
21900    C       NO INTERACTIVE USAGE
22000    C       ************************************************************
22100    C       NO ADDITIONAL ROUTINES USED
22200    C       ************************************************************
22300    C       ************************************************************
22400    C       ************************************************************
22500            SUBROUTINE GERSH
22600    C       ************************************************************
22700    C       COMPUTES GERSHGORIN BANDS FOR CURRENT Q VALUES
22800    C       PLACES RESULT IN COMMON/A5/ BANDS
22900    C       ************************************************************
23000            IMPLICIT COMPLEX(C)
23100            COMMON/A2/D1,D2,D3,ISTART,IEND,ID,MO
23200    C       ************************************************************
23300    C       USES COMMON/A31/,COMMON/A4/,COMMON/A5/ ALSO
23400    C       ************************************************************
23500            COMMON/A31/C1(2,2),C2(2,2),S1(2),S2(2)
23600            COMMON/A4/QR(2,2,201),QI(2,2,201)
23700            COMMON/A5/BAND(2,2,201,2)
23800    C       ************************************************************
23900            DO 30 N=1,IEND
24000            DO 10 I=1,MO
```

```
24100            DO 10 J=1,MO
24200            C2(I,J)=CMPLX(QR(I,J,N),QI(I,J,N))
24300    10      CONTINUE
24400            DO 30 I=1,MO
24500            DO 30 J=1,MO
24600            XR=0.0
24700            XC=0.0
24800            DO 20 K=1,MO
24900            IF(K.NE.J) XR=XR+CABS(C2(I,K))
25000            IF(K.NE.I) XC=XC+CABS(C2(K,J))
25100    20      CONTINUE
25200            BAND(I,J,N,1)=XR
25300            BAND(I,J,N,2)=XC
25400    30      CONTINUE
25500            RETURN
25600            END
25700    C       ***************************************************************
25800    C       NO INTERACTIVE USAGE
25900    C       ***************************************************************
26000    C       NO ADDITIONAL ROUTINES USED
26100    C       ***************************************************************
26200    C       ***************************************************************
26300    C       ***************************************************************
26400            SUBROUTINE OSTROW
26500    C       ***************************************************************
26600    C       COMP. OSTROWSKI BANDS FROM CURRENT GERSH. BANDS,Q AND F VALUES
26700    C       PLACES RESULT IN COMMON/A6/ OST
26800    C       ***************************************************************
26900            IMPLICIT COMPLEX(C)
27000            COMMON/A2/OMEGO,OMEGND,DTOMEG,ISTART,IEND,NP,MO
27100    C       ***************************************************************
27200    C       USES COMMON/A7/,COMMON/A4/,COMMON/A5/,COMMON/A6/ ALSO
27300    C       ***************************************************************
27400            COMMON/A4/QR(2,2,201),QI(2,2,201)
27500            COMMON/A5/BAND(2,2,201,2)
27600            COMMON/A6/OST(2,201,2)
27700            COMMON/A7/XS(2,2),XC(2,2,6,2)
27800    C       ***************************************************************
27900            DO 40 N=1,IEND
28000            XOR=0.0
28100            XOC=0.0
28200            DO 30 I=1,MO
28300            DO 20 J=1,MO
28400            IF(J.EQ.I) GO TO 20
28500            C=CMPLX((XS(J,J)+QR(J,J,N)),QI(J,J,N))
28600            H=CABS(C)
28700            XOR1=BAND(J,J,N,1)/H
28800            XOC1=BAND(J,J,N,2)/H
28900            IF(XOR1.GT.XOR) XOR=XOR1
29000            IF(XOC1.GT.XOC) XOC=XOC1
29100    20      CONTINUE
29200            OST(I,N,1)=XOR
29300            OST(I,N,2)=XOC
29400    30      CONTINUE
29500    40      CONTINUE
29600            RETURN
29700            END
29800    C       ***************************************************************
29900    C       NO INTERACTIVE USAGE
30000    C       ***************************************************************
```

```
30100    C       NO ADDITIONAL ROUTINES USED
30200    C       ************************************************************
30300    C       ************************************************************
30400    C       ************************************************************
30500            SUBROUTINE MMIJ(I,J,XMIN,XMAX,YMIN,YMAX)
30600    C       ************************************************************
30700    C       SEARCHS (I,J) COMPOSIT NYQUIST DIAGRAM FOR (X,Y) MIN/MAX
30800    C       ************************************************************
30900            COMMON/A2/D1,D2,D3,ISTART,IEND,NIDI,MO
31000    C       ************************************************************
31100    C       USES COMMON/A4,COMMON/A5/ ALSO
31200    C       ************************************************************
31300            COMMON/A4/QR(2,2,201),QI(2,2,201)
31400            COMMON/A5/BAND(2,2,201,2)
31500    C       ************************************************************
31600            DO 20 NL=1,IEND
31700            IRC=1
31800            IF(BAND(I,J,NL,1).GE.BAND(I,J,NL,2)) IRC=2
31900            XMIN1=QR(I,J,NL)-BAND(I,J,NL,IRC)
32000            XMAX1=QR(I,J,NL)+BAND(I,J,NL,IRC)
32100            IF(I.NE.J) XMIN1=QR(I,J,NL)
32200            IF(I.NE.J) XMAX1=QR(I,J,NL)
32300            IF(XMIN1.LT.XMIN) XMIN=XMIN1
32400            IF(XMAX1.GT.XMAX) XMAX=XMAX1
32500            YMIN1=QI(I,J,NL)-BAND(I,J,NL,IRC)
32600            YMAX1=QI(I,J,NL)+BAND(I,J,NL,IRC)
32700            IF(I.NE.J) YMIN1=QI(I,J,NL)
32800            IF(I.NE.J) YMAX1=QI(I,J,NL)
32900            IF(YMIN1.LT.YMIN) YMIN=YMIN1
33000            IF(YMAX1.GT.YMAX) YMAX=YMAX1
33100    20      CONTINUE
33200            RETURN
33300            END
33400    C       ************************************************************
33500    C       NO INTERACTIVE USAGE
33600    C       ************************************************************
33700    C       NO ADDITIONAL ROUTINES CALLED
33800    C       ************************************************************
33900    C       ************************************************************
```

Table A.2.2. GRAPH.FOR listing

```
00100   C       ************************************************************
00200   C       ************************************************************
00300           SUBROUTINE PTNY1(I,J)
00400   C       ************************************************************
00500   C       PLOTS (I,J) NYQUIST DIAGRAM
00600   C       ************************************************************
00700           COMMON/A2/D1,D2,D3,ISTART,IEND,ID,MO
00800   C       ************************************************************
00900   C       USES COMMON/A4/ ALSO
01000   C       ************************************************************
01100           COMMON/A4/QR(2,2,201),QI(2,2,201)
01200   C       ************************************************************
01300           DIMENSION XR(202),XI(202)
01400   C       NOTE : FOR PLOTTING MORE THAN 201 POINTS IN Q CHANGE
01500   C              DIMENSIONS ACCORDINGLY
01600           IMANY=IEND-ISTART+1
01700           XR(1)=IMANY
01800           XI(1)=IMANY
01900           DO 10 N=1,IMANY
02000           XR(N+1)=QR(I,J,N+ISTART-1)
02100           XI(N+1)=QI(I,J,N+ISTART-1)
02200   10      CONTINUE
02300           CALL CHECK(XR,XI)
02400           CALL DSPLAY(XR,XI)
02500           RETURN
02600           END
02700   C       ************************************************************
02800   C       NO INTERACTIVE USAGE
02900   C       ************************************************************
03000   C       NO ADDITIONAL ROUTINES USED
03100   C       ************************************************************
03200   C       ************************************************************
03300   C       ************************************************************
03400           SUBROUTINE PTGR1(I,J,IRC)
03500   C       ************************************************************
03600   C       PLOTS GERSHGORIN BANDS OF (I,J) ENTRY OF Q
03700   C       IRC = 1 FOR PLOTTING BANDS BASED ON ROWS
03800   C           = 2 FOR PLOTTING BANDS BASED ON COLUMNS
03900   C       ************************************************************
04000           COMMON/A2/OMEG0,OMEGND,DTOMEG,ISTART,IEND,NP,MO
04100   C       ************************************************************
04200   C       USES COMMON/A4/,COMMON/A5/
04300           COMMON/A4/QR(2,2,201),QI(2,2,201)
04400           COMMON/A5/BAND(2,2,201,2)
04500   C       ************************************************************
04600           DIMENSION X(102),Y(102)
04700           NSTOP=(IEND-ISTART+1)/NP+1
04800           X(1)=101
04900           Y(1)=101
05000           DO 20 N=1,NSTOP
05100           NI=(N-1)*NP+ISTART
05200           DO 10 K=1,101
05300           X(K+1)=QR(I,J,NI)+BAND(I,J,NI,IRC)*COS((K-1)*.0628318)
05400           Y(K+1)=QI(I,J,NI)+BAND(I,J,NI,IRC)*SIN((K-1)*.0628318)
05500   10      CONTINUE
05600           CALL CPLOT(X,Y)
05700   20      CONTINUE
05800           RETURN
05900           END
06000   C       ************************************************************
```

```
06100   C       NO INTERACTIVE USAGE
06200   C       ************************************************************
06300   C       NO ADDITIONAL ROUTINES CALLED
06400   C       ************************************************************
06500   C       ************************************************************
06600   C       ************************************************************
06700           SUBROUTINE PTOST1(I,IRC)
06800   C       ************************************************************
06900   C       PLOTS OSTROWSKI BANDS FOR H(I,I)
07000   C       IRC = 1 FOR BANDS BASED ON ROWS
07100   C           = 2 FOR BANDS BASED ON COLUMNS
07200   C       ************************************************************
07300           COMMON/A2/OMEG0,OMEGND,DTOMEG,ISTART,IEND,NP,MO
07400   C       ************************************************************
07500   C       USES COMMON/A4/,COMMON/A6/ ALSO
07600   C       ************************************************************
07700           COMMON/A4/QR(2,2,201),QI(2,2,201)
07800           COMMON/A6/OST(2,201,2)
07900   C       ************************************************************
08000           DIMENSION X(102),Y(102)
08100           NSTOP=(IEND-ISTART+1)/NP+1
08200           X(1)=101
08300           Y(1)=101
08400           DO 20 N=1,NSTOP
08500           NI=(N-1)*NP+ISTART
08600           DO 10 K=1,101
08700           X(K+1)=QR(I,I,NI)+OST(I,NI,IRC)*COS((K-1)*.0628318)
08800           Y(K+1)=QI(I,I,NI)+OST(I,NI,IRC)*SIN((K-1)*.0628318)
08900   10      CONTINUE
09000           CALL CPLOT(X,Y)
09100   20      CONTINUE
09200           RETURN
09300           END
09400   C       ************************************************************
09500   C       NO INTERACTIVE USAGE
09600   C       ************************************************************
09700   C       NO ADDITIONAL ROUTINES USED
09800   C       ************************************************************
09900   C       ************************************************************
10000   C       ************************************************************
10100           SUBROUTINE PTBIG1
10200   C       ************************************************************
10300   C       PLOTS NYQUIST DIAGRAMS W/OSTROWSKI BANDS
10400   C       USES MAXIMUM SCREEN SIZE
10500   C       ************************************************************
10600   C       NO COMMON BLOCK USAGE
10700   C       ************************************************************
10800   901     FORMAT(I1,X,I1)
10900   10      CONTINUE
11000           CALL ANMODE
11100           CALL HOME
11200           TYPE 900
11300   900     FORMAT(X,'TYPE 0,OR TYPE I,1/2 FOR H(I,I) AND OST ROWS/COLUMS')
11400           ACCEPT 901,I,IRC
11500           IF(I.EQ.0) GO TO 20
11600           CALL BINITT
11700           CALL NEWPAG
11800           CALL SLIMX(75,1000)
11900           CALL SLIMY(50,740)
12000           CALL PTNY1(I,I)
```

```
12100           CALL PTOST1(I,IRC)
12200           GO TO 10
12300    20     CONTINUE
12400           CALL ANMODE
12500           CALL NEWPAG
12600           RETURN
12700           END
12800    C      ************************************************************
12900    C      INTERACTIVELY ACCEPTS:
13000    C              1) DIAGONAL ELEMENT
13100    C              2) BAND BASED ON ROW/COLUMN
13200    C      ************************************************************
13300    C      ADDITIONAL ROUTINES CALLED : PTNY1,PTOST1
13400    C      ************************************************************
13500    C      ************************************************************
13600    C      ************************************************************
13700           SUBROUTINE MAK4(IRC)
13800    C      ************************************************************
13900    C      PLOTS ALL NYQUIST DIAGRAMS FOR (2X2) SYSTEM ON SCREEN
14000    C      ************************************************************
14100           COMMON/A2/01,02,03,ID0,ID1,ID2,MO
14200    C      ************************************************************
14300    C      NO ADDITIONAL COMMON USAGE
14400    C      ************************************************************
14500           XMIN=10000.
14600           YMIN=10000.
14700           XMAX=-10000.
14800           YMAX=-10000.
14900           DO 10 I=1,MO
15000           DO 10 J=1,MO
15100           ID=I
15200           JD=J
15300           CALL MMIJ(ID,JD,XMIN,XMAX,YMIN,YMAX)
15400    10     CONTINUE
15500           CALL DLIMX(XMIN,XMAX)
15600           CALL DLIMY(YMIN,YMAX)
15700           CALL ERASE
15800           CALL PLACE(4)
15900           CALL PTNY1(1,1)
16000           IF(IRC.NE.0) CALL PTGR1(1,1,IRC)
16100           CALL PLACE(5)
16200           CALL PTNY1(1,2)
16300           CALL PLACE(6)
16400           CALL PTNY1(2,1)
16500           CALL PLACE(7)
16600           CALL PTNY1(2,2)
16700           IF(IRC.NE.0) CALL PTGR1(2,2,IRC)
16800           RETURN
16900           END
17000    C      ************************************************************
17100    C      NO INTERACTIVE USAGE
17200    C      ************************************************************
17300    C      ADDITIONAL ROUTINES CALLED : MMIJ,PTNY1,PTGR1
17400    C      ************************************************************
17500    C      ************************************************************
17600    C      ************************************************************
17700           SUBROUTINE ORW
17800    C      ************************************************************
17900    C      PLOT ROUTING ROUTINE
18000    C      ************************************************************
```

```
18100   C       NO COMMON BLOCK USAGE
18200   C       ************************************************************
18300   10      CALL ANMODE
18400           CALL HOME
18500           CALL NEWLIN
18600           CALL NEWLIN
18700           CALL NEWLIN
18800           TYPE 900
18900   900     FORMAT(5X,'PLOT DIAGRAMS? TYPE: 0 FOR NO,1 FOR YES')
19000           ACCEPT *,IDUM
19100           IF(IDUM.EQ.0) GO TO 100
19200           ENTRY DRW1
19300           CALL ANMODE
19400           TYPE 901
19500   901     FORMAT(X,'ENTER -0,0- FOR A,OR -I,J- FOR Q(I,J)')
19600           ACCEPT *,I,J
19700           IF(I.EQ.0.AND.J.EQ.0) GO TO 20
19800           XMIN=100000.
19900           YMIN=100000.
20000           XMAX=-100000.
20100           YMAX=-100000.
20200           CALL MMIJ(I,J,XMIN,XMAX,YMIN,YMAX)
20300           CALL DLIMX(XMIN,XMAX)
20400           CALL DLIMY(YMIN,YMAX)
20500           CALL ERASE
20600           CALL PLACE(1)
20700           CALL PTNY1(I,J)
20800   20      CONTINUE
20900           CALL HOME
21000           CALL NEWLIN
21100           TYPE 902
21200   902     FORMAT(5X,'PLOT BANDS? TYPE: 0 FOR NO,1 FOR ROWS,2 FOR COLUMNS')
21300           ACCEPT *,IRC
21400           IF(IRC.EQ.0.AND.I.NE.0) GO TO 10
21500           IF(I.EQ.0.AND.J.EQ.0) CALL MAK4(IRC)
21600           IF(I.EQ.0.AND.J.EQ.0) GO TO 10
21700           CALL PTGR1(I,J,IRC)
21800           GO TO 10
21900   100     CONTINUE
22000           RETURN
22100           END
22200   C       ************************************************************
22300   C       INTERACTIVELY ACCEPTS :
22400   C               1) PLOT YES/NO
22500   C               2) PLOT 4 DIAGRMS/I,J ELEMENT
22600   C               3) PLOT GERSH.BANDS NO/ROWS/COLUMNS
22700   C       ************************************************************
22800   C       ADITIONAL ROUTINES USED : MMIJ,PTNY1,PTGR1,MAK4
22900   C       ************************************************************
23000   C       ************************************************************
```

4) Subroutine QOUT outputs the numerical value of the open loop system matrix $\hat{Q}(s)$ to the terminal screen or lineprinter.

5) Subroutine QDSKWR interactively stores the open loop system $\hat{Q}(s)$ numerics on computer disk for the next design run.

6) Subroutine QDSKRD automatically reads the open loop system $\hat{Q}(s)$ numerics from disk so they do not have to be recalculated each time the program is run.

7) Subroutine OUT interactively outputs system matrices and programming variable to the terminal screen or lineprinter.

8) Subroutine FINISH notifies the user he is stopping, causes system outputs and disk storage to possibly be performed, and terminates program execution.

A listing of this program is given in Table A.2.3.

IV. AUTOMP.FOR contains the main program used in investigating the autopilot program.

V. OLISP.FOR contains the open loop constant input compensator design package.

VI. IFDCP.FOR contains the open loop frequency dependent compensator design package.

VII. CLSP.FOR contains the closed loop feedback design package.

VIII. DATA12.FOR contains the subroutine defining the system and programming data for the autopilot problem.

IX. QINV2.FOR contains the subroutine which takes the point by point inverse of the autopilot plant data.

See Table A.2.4 for a program listing.

Table A.2.3. AUX.FOR Listing

```
00100    C       ****************************************************************
00200    C       ****************************************************************
00300            SUBROUTINE MODIA2
00400    C       ****************************************************************
00500    C       MODIFIES INTEGER VALUES IN COMMON/A2/
00600    C       ****************************************************************
00700            COMMON/A2/OMEGO,OMEGND,DTOMEG,ISTART,IEND,NP,MO
00800    C       ****************************************************************
00900    C       NO OTHER COMMON BLOCK USAGE
01000    C       ****************************************************************
01100            TYPE 900
01200    900     FORMAT(X,'ENTER ISTART,IEND,NP,OR =,= AND RETURN')
01300            ACCEPT *,I1,I2,I3
01400            IF(I1.NE.0) ISTART=I1
01500            IF(I2.NE.0) IEND=I2
01600            IF(I3.NE.0) NP=I3
01700            TYPE *,ISTART,IEND,NP
01800            RETURN
01900            END
02000    C       ****************************************************************
02100    C       INTERACTIVE USAGE
02200    C               1) ENTER VALUES OR COMMA
02300    C       ****************************************************************
02400    C       NO OTHER ROUTINES USES
02500    C       ****************************************************************
02600    C       ****************************************************************
02700    C       ****************************************************************
02800            SUBROUTINE MODXS
02900    C       ****************************************************************
03000    C       MODIFIES AND OUTPUTS MATRIX XS
03100    C       ****************************************************************
03200            COMMON/A2/D1,D2,D3,ID0,ID1,ID2,MO
03300    C       ****************************************************************
03400    C       USES COMMON/A7/ ALSO
03500    C       ****************************************************************
03600            COMMON/A7/XS(2,2),XC(2,2,6,2)
03700    C       ****************************************************************
03800            CALL ANMODE
03900    10      TYPE 901
04000    901     FORMAT(X,'MODIFY XS? TYPE: 0 FOR NO,1 FOR YES')
04100            ACCEPT *,IDUM
04200            IF(IDUM.EQ.0) GO TO 20
04300            TYPE 904
04400    904     FORMAT(X,'TYPE:  I,J,X IN XS(I,J)=X')
04500            ACCEPT *,I,J,X
04600            XS(I,J)=X
04700            GO TO 10
04800    20      TYPE 902
04900    902     FORMAT(X,'OUTPUT XS? TYPE:  0 FOR NO,1 FOR TTY,2 FOR LP')
05000            ACCEPT *,IDUM
05100            IF(IDUM.EQ.0) GO TO 40
05200            DO 30 I=1,MO
05300            DO 30 J=1,MO
05400            IF(IDUM.EQ.1) TYPE 903,I,J,XS(I,J)
05500            IF(IDUM.EQ.2) PRINT 903,I,J,XS(I,J)
05600    903     FORMAT(X,'XS(',I1,',',I1,') = ',F7.2)
05700    30      CONTINUE
05800    40      CONTINUE
05900            RETURN
06000            END
```

```
06100   C       ****************************************************************
06200   C       INTERACTIVE USAGE:
06300   C               1) MODIFY XS YES/NO
06400   C               2) ENTER VALUES
06500   C               3) OUTPUT XS NO/TTY/LP
06600   C       ****************************************************************
06700   C       NO ADDITIONAL ROUTINES USED
06800   C       ****************************************************************
06900   C       ************************************************************
07000   C       ************************************************************
07100           SUBROUTINE MODXC
07200   C       ************************************************************
07300   C       MODIFIES AND OUTPUTS MATRIX XC
07400   C       ************************************************************
07500           COMMON/A2/D1,D2,D3,ID0,ID1,ID2,MO
07600   C       ************************************************************
07700   C       USES COMMON/A7/ ALSO
07900   C       ************************************************************
07900           COMMON/A7/XS(2,2),XC(2,2,6,2)
08000   C       ************************************************************
08100           CALL ANMODE
08200   10      TYPE 901
08300   901     FORMAT(X,'MODIFY XC? TYPE: 0 FOR NO, 1 FOR YES')
08400           ACCEPT *,IDUM
08500           IF(IDUM.EQ.0) GO TO 20
08600           TYPE 902
08700   902     FORMAT(X,'I,J,N,X,1/2  IN  XC(I,J)=X*(S)**N, NUM/DOM')
08800           ACCEPT *,I,J,N,X,ND
08900           XC(I,J,N+1,ND)=X
09000           GO TO 10
09100   20      CONTINUE
09200           TYPE 903
09300   903     FORMAT(X,'OUTPUT XC? TYPE: 0 FOR NO, 1 FOR TTY, 2 FOR LP')
09400           ACCEPT *,IDUM
09500           IF(IDUM.EQ.0) GO TO 60
09600           IF(IDUM.EQ.2) GO TO 40
09700           DO 30 I=1,MO
09800           DO 30 J=1,MO
09900           TYPE 904,XC(I,J,3,1),XC(I,J,2,1),XC(I,J,1,1)
10000   904     FORMAT(11X,F7.2,3X,F7.2,3X,F7.2)
10100           TYPE 905,I,J
10200   905     FORMAT(X,'XC(',I1,',',I1,') = ',29('-'))
10300           TYPE 904,XC(I,J,3,2),XC(I,J,2,2),XC(I,J,1,2)
10400           TYPE 906
10500   906     FORMAT(X,' ')
10600   30      CONTINUE
10700   40      IF(IDUM.NE.2)GO TO 60
10800           DO 50 I=1,MO
10900           DO 50 J=1,MO
11000           PRINT 904,XC(I,J,3,1),XC(I,J,2,1),XC(I,J,1,1)
11100           PRINT 905,I,J
11200           PRINT 904,XC(I,J,3,2),XC(I,J,2,2),XC(I,J,1,2)
11300           PRINT 906
11400   50      CONTINUE
11500   60      CONTINUE
11600           RETURN
11700           END
11800   C       ************************************************************
11900   C       INTERACTIVE USAGE
12000   C               1) MODIFY XC YES/NO
```

```
C               2) ENTER VALUES
C               3) OUTPUT XC NO/TTY/LP
C       ************************************************************
C       NO ADDITIONAL ROUTINES USED
C       ************************************************************
C       ****************************************************************
C       ****************************************************************
      SUBROUTINE QOUT(IOUT)
C       ****************************************************************
C       OUTPUTS CURRENT Q VALUES IN COMMON/A4/
C       IOUT = 1 FOR TTY
C            = 2 FOR LP
C       ****************************************************************
      IMPLICIT COMPLEX(C)
      COMMON/A2/OMEG0,OMEGNO,DTOMEG,ISTART,IEND,NP,MO
C       ****************************************************************
C       USES COMMON/A4/ ALSO
C       ****************************************************************
      COMMON/A4/QR(2,2,201),QI(2,2,201)
C       ****************************************************************
      DO 20 I=1,MO
      DO 20 J=1,MO
      DO 10 N=ISTART,IEND
      IF(IOUT.EQ.1) TYPE *,I,J,N,QR(I,J,N),QI(I,J,N)
      IF(IOUT.EQ.2) PRINT *,I,J,N,QR(I,J,N),QI(I,J,N)
10    CONTINUE
20    CONTINUE
      RETURN
      END
C       ****************************************************************
C       NO INTERACTIVE USAGE
C       ****************************************************************
C       NO ADDITIONAL ROUTINES USED
C       ****************************************************************
C       ****************************************************************
C       ****************************************************************
      SUBROUTINE QDSKWR
C       ****************************************************************
C       WRITES Q DATA TO DISK FILE Q1.DAT
C       ****************************************************************
      COMMON/A2/D1,D2,D3,ISTART,IEND,ID1,MO
C       ****************************************************************
C       USES COMMON/A4/ ALSO
C       ****************************************************************
      COMMON/A4/QR(2,2,201),QI(2,2,201)
C       ****************************************************************
      TYPE 888
888   FORMAT(X,'STORE Q VALUES? TYPE 0 FOR NO, 1 FOR YES')
      ACCEPT *,IDO
      IF(IDO.EQ.0) GO TO 40
      IDUM=1
      OPEN(UNIT=1,ACCESS='SEQOUT',FILE='Q1.DAT')
      WRITE(1,900) IDUM
900   FORMAT(X,I1)
      DO 30 N=1,IEND
      DO 30 I=1,MO
      DO 30 J=1,MO
      WRITE(1,901) I,J,N,QR(I,J,N),QI(I,J,N)
901   FORMAT(X,I1,X,I1,X,I3,X,E14.7,X,E14.7)
30    CONTINUE
```

```
18100            CLOSE(UNIT=1,ACCESS='SEQOUT',FILE='Q1.DAT')
18200    40      CONTINUE
18300            RETURN
18400            END
18500    C       ***************************************************************
18600    C       INTERACTIVE USAGE:
18700    C               1) STORE Q VALUES NO/YES
18800    C       ***************************************************************
18900    C       NO ADDITIONAL ROUTINES USED
19000    C       ***************************************************************
19100    C       ***************************************************************
19200    C       ***************************************************************
19300            SUBROUTINE QDSKRD(IDUM)
19400    C       ***************************************************************
19500    C       READS Q DATA FROM DISK FILE Q1.DAT
19600    C       IDUM = 0 ON RETURN FOR NO PREVIOUSLY STORED DATA
19700    C            = 1 ON RETURN IF Q VALUES ARE READ FROM DISK
19800    C       ***************************************************************
19900            COMMON/A2/D1,D2,D3,ISTART,IEND,ID1,MO
20000    C       ***************************************************************
20100    C       USES COMMON/A4/ ALSO
20200    C       ***************************************************************
20300            COMMON/A4/QR(2,2,201),QI(2,2,201)
20400    C       ***************************************************************
20500    900     FORMAT(X,I1)
20600    901     FORMAT(X,I1,X,I1,X,I3,X,E14.7,X,E14.7)
20700            OPEN(UNIT=1,ACCESS='SEQIN',FILE='Q1.DAT')
20800            READ(1,900) IDUM
20900            IF(IDUM.EQ.0) GO TO 30
21000    10      READ(1,901,END=20) I,J,N,SR,SI
21100            QR(I,J,N)=SR
21200            QI(I,J,N)=SI
21300            GO TO 10
21400    20      CONTINUE
21500            IEND=N
21600    30      CONTINUE
21700            CLOSE(UNIT=1,ACCESS='SEQIN',FILE='Q1.DAT')
21800            RETURN
21900            END
22000    C       ***************************************************************
22100    C       NO INTERACTIVE USAGE
22200    C       ***************************************************************
22300    C       NO ADDITIONAL ROUTINES USED
22400    C       ***************************************************************
22500    C       ***************************************************************
22600    C       ***************************************************************
22700            SUBROUTINE OUT
22800    C       ***************************************************************
22900    C       OUTPUTS COMMON/A1/ VALUES
23000    C       ***************************************************************
23100            COMMON/A2/OMEG0,OMEGND,DTOMEG,ISTART,IEND,NP,MO
23200    C       ***************************************************************
23300    C       USES COMMON/A1/ ALSO
23400    C       ***************************************************************
23500            COMMON/A1/G(2,2,6,2),XKB(2,2,3,2),XLB(2,2,3,2),
23600            1 XKA(2,2),XLA(2,2),F(2,2)
23700    C       ***************************************************************
23800    900     FORMAT(X,'KA(',I1,',',I1,') = ',F7.2,3X,'LA(',I1,',',I1,
23900           /') = ',F7.2,3X,'F(',I1,',',I1,') = ',F7.2)
24000    901     FORMAT(10X,F7.2,3X,F7.2,3X,F7.2,3X,F7.2,3X,F7.2,3X,F7.2)
```

```
24100    902     FORMAT(X,'G(',I1,',',I1,') = ',50('-'))
24200    903     FORMAT(11X,F7.2,3X,F7.2,3X,F7.2)
24300    904     FORMAT(X,'KB(',I1,',',I1,') = ',32('-'))
24400    905     FORMAT(X,'LB(',I1,',',I1,') = ',32('-'))
24500    906     FORMAT(X)
24600    907     FORMAT(X,'OMEGO = ',F7.2,X,'OMEGND = ',F7.2,X,'DTOMEG = '
24700          /  ,F7.2,X,'IEND = ',I3,X,'NP = ',I3)
24800    908     FORMAT(X,'THE ORDER OF THE SYSTEM IS = ',I1)
24900    909     FORMAT(X,'FOR OUTPUT TYPE:')
25000    910     FORMAT(X,' 0 FOR NONE,1 FOR TTY,2 FOR LP')
25100    911     FORMAT(X,'TYPE: 0 TO CONTINUE, TYPE: 1 TO DO AGAIN')
25200    912     FORMAT(I1)
25300    913     FORMAT(X,'FOR SYSTEM OUTPUT TYPE 0, FOR Q OUTPUT TYPE 1')
25400    5       TYPE 909
25500            TYPE 910
25600            ACCEPT *,I1
25700            IF(I1.EQ.0) RETURN
25800            TYPE 906
25900            TYPE 913
26000            ACCEPT *,I2
26100            IF(I2.EQ.1) CALL QOUT(I1)
26200            IF(I2.EQ.1) GO TO 90
26300            I1=I1-1
26400            IF(I1.EQ.1) GO TO 100
26500            DO 10 I=1,MO
26600            DO 10 J=1,MO
26700            TYPE  906
26800            TYPE  901,G(I,J,5,1),G(I,J,4,1),G(I,J,3,1),G(I,J,2,1),G(I,J,1,1)
26900            TYPE  902, I,J
27000            TYPE  901,G(I,J,5,2),G(I,J,4,2),G(I,J,3,2),G(I,J,2,2),G(I,J,1,2)
27100            TYPE  906
27200    10      CONTINUE
27300            DO 20 I=1,MO
27400            DO 20 J=1,MO
27500            TYPE  906
27600            TYPE  900,I,J,XKA(I,J),I,J,XLA(I,J),I,J,F(I,J)
27700    20      CONTINUE
27800            DO  30 I=1,MO
27900            DO  30 J=1,MO
28000            TYPE  906
28100            TYPE  903,XKB(I,J,3,1),XKB(I,J,2,1),XKB(I,J,1,1)
28200            TYPE  904,I,J
28300            TYPE  903,XKB(I,J,3,2),XKB(I,J,2,2),XKB(I,J,1,2)
28400            TYPE  906
28500    30      CONTINUE
28600            DO  40 I=1,MO
28700            DO  40 J=1,MO
28800            TYPE  906
28900            TYPE  903,XLB(I,J,3,1),XLB(I,J,2,1),XLB(I,J,1,1)
29000            TYPE  905,I,J
29100            TYPE  903,XLB(I,J,3,2),XLB(I,J,2,2),XLB(I,J,1,2)
29200            TYPE  906
29300    40      CONTINUE
29400    45      TYPE  907,OMEGO,OMEGND,DTOMEG,IEND,NP
29500            TYPE 908,MO
29600    90      CONTINUE
29700            TYPE 911
29800            ACCEPT 912,IDUM
29900            IF(IDUM.EQ.1) GO TO 5
30000            IF(I1.EQ.0) RETURN
```

```
30100   100     CONTINUE
30200           DO 110 I=1,MO
30300           DO 110 J=1,MO
30400           PRINT  906
30500           PRINT 901,G(I,J,5,1),G(I,J,4,1),G(I,J,3,1),G(I,J,2,1),G(I,J,1,1)
30600           PRINT 902, I,J
30700           PRINT 901,G(I,J,5,2),G(I,J,4,2),G(I,J,3,2),G(I,J,2,2),G(I,J,1,2)
30800           PRINT  906
30900    110    CONTINUE
31000           DO 120 I=1,MO
31100           DO 120 J=1,MO
31200           PRINT  906
31300           PRINT  900,I,J,XKA(I,J),I,J,XLA(I,J),I,J,F(I,J)
31400    120    CONTINUE
31500           DO  130 I=1,MO
31600           DO  130 J=1,MO
31700           PRINT  906
31800           PRINT  903,XKB(I,J,3,1),XKB(I,J,2,1),XKB(I,J,1,1)
31900           PRINT  904,I,J
32000           PRINT  903,XKB(I,J,3,2),XKB(I,J,2,2),XKB(I,J,1,2)
32100           PRINT  906
32200    130    CONTINUE
32300           DO  140 I=1,MO
32400           DO  140 J=1,MO
32500           PRINT  906
32600           PRINT  903,XLB(I,J,3,1),XLB(I,J,2,1),XLB(I,J,1,1)
32700           PRINT  905,I,J
32800           PRINT  903,XLB(I,J,3,2),XLB(I,J,2,2),XLB(I,J,1,2)
32900           PRINT  906
33000    140    CONTINUE
33100    145    PRINT  907,OMEG0,OMEGND,DTOMEG,IEND,NP
33200           PRINT 908,MO
33300           TYPE 911
33400           ACCEPT *,IDUM
33500           IF(IDUM.EQ.1) GO TO 5
33600           RETURN
33700           END
33800   C       ***************************************************************
33900   C       INTERACTIVE USAGE
34000   C               1) OUTPUT DESIRED NONE/TTY/LP
34100   C               2) OUTPUT SYSTEM/Q ?
34200   C               3) DO AGAIN NO/YES
34300   C       ***************************************************************
34400   C       NO ADDITIONAL ROUTINES USED
34500   C       ***************************************************************
34600   C       ***************************************************************
34700   C       ***************************************************************
34800           SUBROUTINE FINISH
34900   C       ***************************************************************
35000   C       STOPS EXECUTION OF THE MAIN PROGRAM
35100   C       ***************************************************************
35200   C       NO COMMON BLOCK USAGE
35300   C       ***************************************************************
35400           CALL ANMODE
35500           TYPE 940
35600   940     FORMAT(X,'YOU ARE DONE!BEST PRINT OUT SYSTEM!!!')
35700           CALL OUT
35800           CALL NEWPAG
35900           CALL ODSKWR
36000           CALL FINITT(0,700)
```

```
36100           STOP
36200           RETURN
36300           END
36400     C     ************************************************************************
36500     C     NO INTERACTIVE USAGE
36600     C     ************************************************************************
36700     C     ADDITIONAL ROUTINES CALLED:OUT,QDSKWR
36800     C     ************************************************************************
```

Table A.2.4. Autopilot Design Program

```
00100   C
00200   C
00300   C       ****************************************************************
00400   C
00500   C       THIS IS THE MAIN PROGRAM FOR THE AUTOPILOT DESIGN
00600   C
00700   C       ****************************************************************
00800           IMPLICIT COMPLEX(C)
00900           COMMON/A1/G(2,2,6,2),XKB(2,2,3,2),XLB(2,2,3,2),
01000           1 XKA(2,2),XLA(2,2),F(2,2)
01100           COMMON/A2/OMEG0,OMEGND,DTOMEG,ISTART,IEND,NP,MO
01200           COMMON/A31/C1(2,2),C2(2,2),S1(2),S2(2)
01300           COMMON/A32/XR1(2,2,201),XI1(2,2,201)
01400           COMMON/A33/XR2(2,2,201),XI2(2,2,201)
01500           COMMON/A4/QR(2,2,201),QI(2,2,201)
01600           COMMON/A5/BAND(2,2,201,2)
01700           COMMON/A6/DST(2,201,2)
01800           COMMON/A7/XS(2,2),XC(2,2,6,2)
01900   C       ****************************************************************
02000           CALL INITT(120)
02100           CALL BINITT
02200           CALL DATA12
02300           IEND=(OMEGND-OMEG0)/DTOMEG+1
02400           IF(IEND.GT.201) IEND=201
02500           CALL QDSKRD(IDUM)
02600           IF(IDUM.EQ.1) TYPE 888
02700   888     FORMAT(X,'Q VALUES WERE OBTAINED FROM DISK')
02800           IF(IDUM.EQ.1) GO TO 10
02900           DO 5 I=1,MO
03000           DO 5 J=1,MO
03100           XS(I,J)=0.0
03200           IF(I.EQ.J) XS(I,J)=1.0
03300   5       CONTINUE
03400           CALL MAP(0)
03500           CALL MIMP
03600           CALL QINV
03700   10      CONTINUE
03800           TYPE 900
03900   900     FORMAT(X,'TYPE: 0 TO STOP')
04000           TYPE 901
04100   901     FORMAT(X,'TYPE: 1 FOR OPEN LOOP PACKAGE')
04200           TYPE 902
04300   902     FORMAT(X,'TYPE: 2 FOR INPUT COMPENSATOR PACKAGE')
04400           TYPE 903
04500   903     FORMAT(X,'TYPE: 3 FOR CLOSED LOOP PACKAGE')
04600           TYPE 904
04700   904     FORMAT(X,'TYPE: 4 FOR OUTPUT')
04800           TYPE 905
04900   905     FORMAT(X,'TYPE: 5 TO MODIFY INTEGER COMMON/A2/ VALUES')
05000           ACCEPT *,IDUM
05100           IF(IDUM.EQ.0) CALL FINISH
05200           IF(IDUM.EQ.1) CALL OLISP
05300           IF(IDUM.EQ.2) CALL IFDCP
05400           IF(IDUM.EQ.3) CALL CLSP
05500           IF(IDUM.EQ.4) CALL OUT
05600           IF(IDUM.EQ.5) CALL MODIA2
05700           GO TO 10
05800           STOP
05900           END
06000   C       ****************************************************************
```

```
00100  C       ******************************************************************
00200          SUBROUTINE OLISP
00300  C       ******************************************************************
00400          IMPLICIT COMPLEX(C)
00500          COMMON/A1/G(2,2,6,2),XKB(2,2,3,2),XLB(2,2,3,2),
00600          1 XKA(2,2),XLA(2,2),F(2,2)
00700          COMMON/A2/OMEGØ,OMEGND,DTOMEG,ISTART,IEND,NP,MO
00800          COMMON/A31/C1(2,2),C2(2,2),S1(2),S2(2)
00900          COMMON/A32/XR1(2,2,201),XI1(2,2,201)
01000          COMMON/A33/XR2(2,2,201),XI2(2,2,201)
01100          COMMON/A4/QR(2,2,201),QI(2,2,201)
01200          COMMON/A5/BAND(2,2,201,2)
01300          COMMON/A6/DST(2,201,2)
01400          COMMON/A7/XS(2,2),XC(2,2,6,2)
01500  C       ******************************************************************
01600          DO 110 I=1,MO
01700          DO 110 J=1,MO
01800          XS(I,J)=XKA(I,J)
01900          DO 100 N=1,IEND
02000          XR1(I,J,N)=QR(I,J,N)
02100          XI1(I,J,N)=QI(I,J,N)
02200  100     CONTINUE
02300  110     CONTINUE
02400          CALL GERSH
02500          TYPE 910
02600  910     FORMAT(X,'DEALING W/ OPEN LOOP INPUT CONST. COMP')
02700  120     CONTINUE
02800          TYPE 900
02900  900     FORMAT(X,'TYPE: 0 TO RETURN, 1 TO PLOT, 2 TO MODIFY')
03000          ACCEPT *,IDUM
03100          CALL NEWPAG
03200          IF(IDUM.EQ.0) GO TO 140
03300          IF(IDUM.EQ.1) GO TO 130
03400          CALL MODXS
03500          GO TO 120
03600  130     CONTINUE
03700          CALL MIMP
03800          CALL GERSH
03900          CALL DRW1
04000          GO TO 120
04100  140     CONTINUE
04200          DO 150 I=1,MO
04300          DO 150 J=1,MO
04400          XKA(I,J)=XS(I,J)
04500  150     CONTINUE
04600  199     CONTINUE
04700          RETURN
04800          END
```

```
00100   C       ****************************************************************
00200           SUBROUTINE IFDCP
00300   C       ****************************************************************
00400           IMPLICIT COMPLEX(C)
00500           COMMON/A1/G(2,2,6,2),XKB(2,2,3,2),XLB(2,2,3,2),
00600           1 XKA(2,2),XLA(2,2),F(2,2)
00700           COMMON/A2/OMEG0,OMEGND,DTOMEG,ISTART,IEND,NP,MO
00800           COMMON/A31/C1(2,2),C2(2,2),S1(2),S2(2)
00900           COMMON/A32/XR1(2,2,201),XI1(2,2,201)
01000           COMMON/A33/XR2(2,2,201),XI2(2,2,201)
01100           COMMON/A4/QR(2,2,201),QI(2,2,201)
01200           COMMON/A5/BAND(2,2,201,2)
01300           COMMON/A6/OST(2,201,2)
01400           COMMON/A7/XS(2,2),XC(2,2,6,2)
01500   C       ****************************************************************
01600           DO 250 I=1,MO
01700           DO 250 J=1,MO
01800           DO 210 K=1,IEND
01900           XR2(I,J,K)=QR(I,J,K)
02000           XI2(I,J,K)=QI(I,J,K)
02100   210     CONTINUE
02200           DO 240 L=1,2
02300           DO 220 K=1,3
02400           XC(I,J,K,L)=XKB(I,J,K,L)
02500   220     CONTINUE
02600           DO 230 K=4,6
02700           XC(I,J,K,L)=0.0
02800   230     CONTINUE
02900   240     CONTINUE
03000   250     CONTINUE
03100           CALL GERSH
03200           TYPE 920
03300   920     FORMAT(X,'DEALING W/ FREQ,DEPNOT, INPUT COMP')
03400   260     CONTINUE
03500           TYPE 900
03600   900     FORMAT(X,'TYPE: 0 TO RETURN, 1 TO PLOT, 2 TO MODIFY')
03700           ACCEPT *,IDUM
03800           CALL NEWPAG
03900           IF(IDUM.EQ.0) GO TO 280
04000           IF(IDUM.EQ.1) GO TO 270
04100           CALL MODXC
04200           GO TO 260
04300   270     CONTINUE
04400           CALL MAG
04500           CALL MCMP
04600           CALL GERSH
04700           CALL DRW1
04800           GO TO 260
04900   280     CONTINUE
05000           DO 290 I=1,MO
05100           DO 290 J=1,MO
05200           DO 290 K=1,3
05300           DO 290 L=1,2
05400           XKB(I,J,K,L)=XC(I,J,K,L)
05500   290     CONTINUE
05600   299     CONTINUE
05700           RETURN
05800           END
05900   C       ****************************************************************
```

```
00100  C       ******************************************************************
00200          SUBROUTINE CLSP
00300  C       ******************************************************************
00400          IMPLICIT COMPLEX(C)
00500          COMMON/A1/G(2,2,6,2),XKB(2,2,3,2),XLB(2,2,3,2),
00600          1 XKA(2,2),XLA(2,2),F(2,2)
00700          COMMON/A2/OMEGO,OMEGND,DTOMEG,ISTART,IEND,NP,MO
00800          COMMON/A31/C1(2,2),C2(2,2),S1(2),S2(2)
00900          COMMON/A32/XR1(2,2,201),XI1(2,2,201)
01000          COMMON/A33/XR2(2,2,201),XI2(2,2,201)
01100          COMMON/A4/OR(2,2,201),OI(2,2,201)
01200          COMMON/A5/BAND(2,2,201,2)
01300          COMMON/A6/OST(2,201,2)
01400          COMMON/A7/XS(2,2),XC(2,2,6,2)
01500  C       ******************************************************************
01600          DO 310 I=1,MO
01700          DO 310 J=1,MO
01800          XS(I,J)=F(I,J)
01900  310     CONTINUE
02000          CALL GERSH
02100          CALL OSTROW
02200          TYPE 930
02300  930     FORMAT(X,'DEALING W/ CLOSED LOOP SYSTEM')
02400  320     CONTINUE
02500          TYPE 900
02600  900     FORMAT(X,'TYPE: 0 TO RETURN, 1 TO PLOT, 2 TO MODIFY')
02700          ACCEPT *, IDUM
02800          CALL NEWPAG
02900          IF(IDUM.EQ.0) GO TO 340
03000          IF(IDUM.EQ.1) GO TO 330
03100          CALL MODXS
03200          GO TO 320
03300  330     CONTINUE
03400          CALL OSTROW
03500          CALL PTBIG1
03600          GO TO 320
03700  340     CONTINUE
03800          DO 350 I=1,MO
03900          DO 350 J=1,MO
04000          F(I,J)=XS(I,J)
04100  350     CONTINUE
04200  399     CONTINUE
04300          RETURN
04400          END
04500  C       ******************************************************************
```

```
00100   C       ************************************************************
00200           SUBROUTINE DATA12
00300   C       ************************************************************
00400           COMMON/A1/G(2,2,6,2),XKB(2,2,3,2),XLB(2,2,3,2),
00500           1 XKA(2,2),XLA(2,2),F(2,2)
00600           COMMON/A2/OMEGO,OMEGND,DTOMEG,ISTART,IEND,NP,MO
00700   C       SETS COMMON BLOCK /A2/ VALUES
00800           OMEGO=0.0
00900           OMEGND=0.16
01000           DTOMEG=0.0008
01100           ISTART=1
01200           IEND=201
01300           NP=10
01400           MO=2
01500   C       SETS ALL DEMON. OF G(S)=CHAR. POLYNOMIAL
01600           DO 10 I=1,MO
01700           DO 10 J=1,MO
01800           G(I,J,1,2)=0.0
01900           G(I,J,2,2)=-.00166
02000           G(I,J,3,2)=.365
02100           G(I,J,4,2)=.364
02200           G(I,J,5,2)=.829
02300           G(I,J,6,2)=1.0
02400   10      CONTINUE
02500   C       SETS INDIVIDUAL NUMERATOR ENTRIES OF G(S)
02600           G(1,1,1,1)=-.0195
02700           G(1,1,2,1)=-.00893
02800           G(1,1,3,1)=-.3121
02900           G(1,1,4,1)=-.3807
03000           G(1,1,5,1)=0.0
03100           G(1,1,6,1)=0.0
03200           G(1,2,1,1)=0.0
03300           G(1,2,2,1)=-.447
03400           G(1,2,3,1)=-.0503
03500           G(1,2,4,1)=-.0404
03600           G(1,2,5,1)=0.0
03700           G(1,2,6,1)=0.0
03800           G(2,1,1,1)=.022635
03900           G(2,1,2,1)=-.000588
04000           G(2,1,3,1)=-.0178
04100           G(2,1,4,1)=.06715
04200           G(2,1,5,1)=0.0
04300           G(2,1,6,1)=0.0
04400           G(2,2,1,1)=0.0
04500           G(2,2,2,1)=.602
04600           G(2,2,3,1)=.06237
04700           G(2,2,4,1)=1.587
04800           G(2,2,5,1)=0.0
04900           G(2,2,6,1)=0.0
05000   C       SETS XKA=I
05100           XKA(1,1)=1.0
05200           XKA(1,2)=0.0
05300           XKA(2,1)=0.0
05400           XKA(2,2)=1.0
05500           RETURN
05600           END
05700   C       ************************************************************
```

```
00100   C       ***************************************************************
00200           SUBROUTINE QINV
00300   C       ***************************************************************
00400   C       TAKES 'LONG HAND' INVERSE OF Q
00500   C       FOR A 2 X 2 SYSTEM ONLY
00600   C       ***************************************************************
00700           IMPLICIT COMPLEX(C)
00800           COMMON/A2/OMEG0,OMEGND,DTOMEG,ISTART,IEND,NP,MO
00900           COMMON/A3/C1(2,2),C2(2,2),S1(2),S2(2)
01000           COMMON/A4/QR(2,2,201),QI(2,2,201)
01100   C       ***************************************************************
01200           DO 30 N=1,IEND
01300           DO 10 I=1,2
01400           DO 10 J=1,2
01500           C1(I,J)=CMPLX(QR(I,J,N),QI(I,J,N))
01600   10      CONTINUE
01700           XREAL=QR(1,1,N)*QR(2,2,N)+QI(1,2,N)*QI(2,1,N)
01800           1 -QI(1,1,N)*QI(2,2,N)-QR(1,2,N)*QR(2,1,N)
01900           XIMAG=QI(1,1,N)*QR(2,2,N)+QR(1,1,N)*QI(2,2,N)
02000           1 -QR(1,2,N)*QI(2,1,N)-QI(1,2,N)*QR(2,1,N)
02100           CCC=CMPLX(XREAL,XIMAG)
02200           IF(CCC.EQ.CMPLX(0.0,0.0)) TYPE 900,N
02300   900     FORMAT(X,'Q SINGULAR UPON INVERSION @ ',I3)
02400           IF(CCC.EQ.CMPLX(0,0,0,0)) GO TO 15
02500           C2(1,1)=C1(2,2)/CCC
02600           C2(1,2)=-C1(1,2)/CCC
02700           C2(2,1)=-C1(2,1)/CCC
02800           C2(2,2)=C1(1,1)/CCC
02900   15      CONTINUE
03000           DO 20 I=1,2
03100           DO 20 J=1,2
03200           QR(I,J,N)=REAL(C2(I,J))
03300           QI(I,J,N)=AIMAG(C2(I,J))
03400   20      CONTINUE
03500   30      CONTINUE
03600           RETURN
03700           END
03800   C       ***************************************************************
```